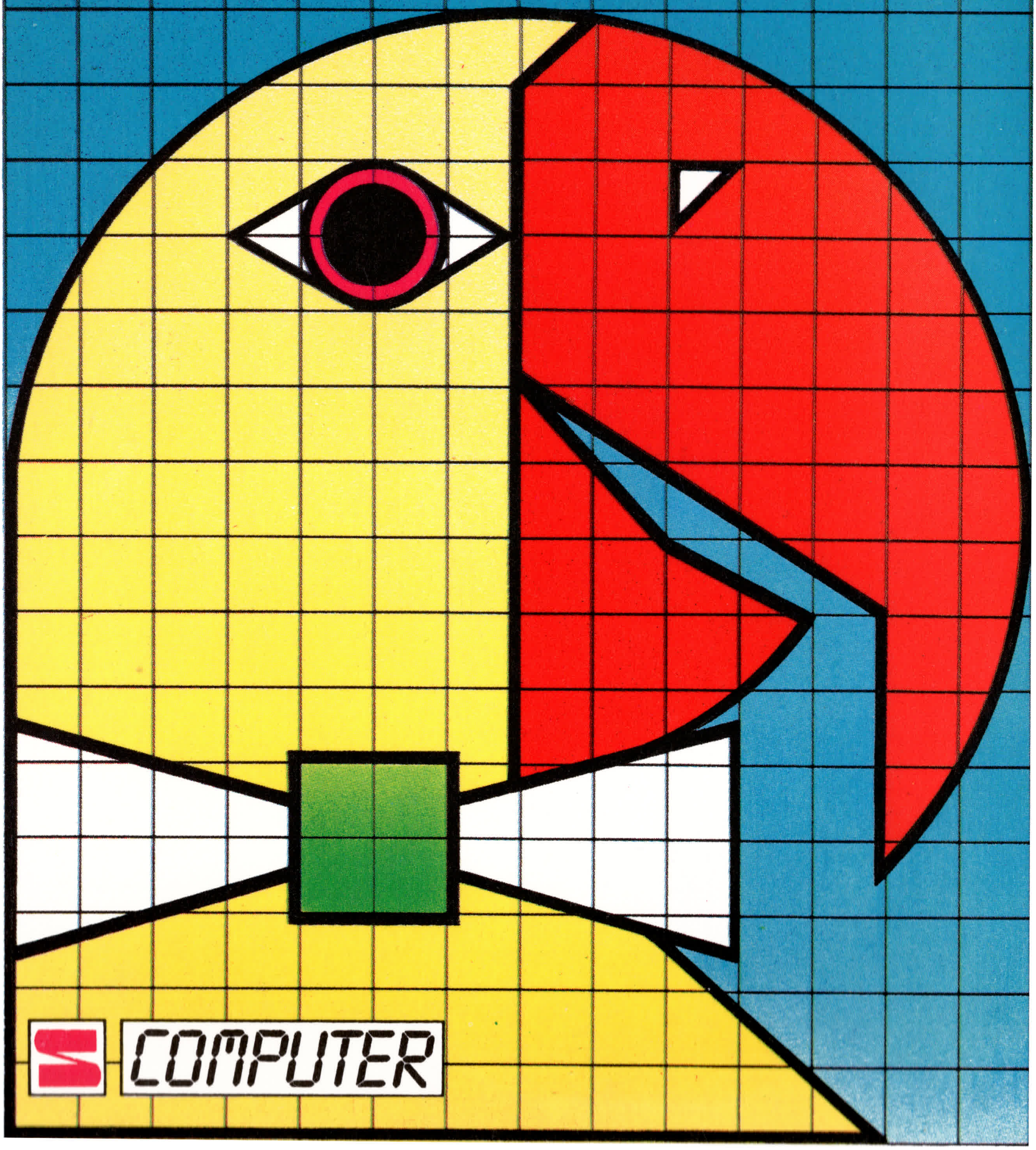
Computer kinderleicht

Programmiere selbst – in BASIC!



Computer kinderleicht

Programmiere selbst – in BASIC!

Georg Seeßlen
Peter Fritzsche

Zeichnungen von Otto Wamser

S COMPUTER

Inhalt

Vorwort 5

Wir unterhalten uns mit dem Computer 7

INPUT oder wie schütze ich mein Programm
vor unbekömmlichem Datenfutter? 29

Ein Ladeprogramm mit Überraschungen und
der C 64 als Quasselkasten 45

Programmieren mit Plan 64

Dem Profi über die Schulter geschaut 78

Ein Musterbeispiel:
Zoll in Zentimeter umrechnen 87

Die elektronische Liebeserklärung 100

Anhang

Was heißt was in BASIC 120

Tips & Tricks für das Eingeben der Listings 124

Liste der Commodore-Steuerzeichen 125

Gebrauchsanleitungen und Listings der
Programme von Band 3

- RENUMBER 128
- Vokabelpaukprogramm 133
- Computerliebe 155
- Leerformular für Listings 156

Vorwort

Programmieren – was ist das eigentlich?
Wer selbständig und nach eigenen Vorstellungen kleine oder größere Programme entwerfen will, muß sich mit zwei Dingen vertraut machen:

1. Mit der "Sprache" des Computers, also mit den Befehlen, Anweisungen und Funktionen, aus denen zum Beispiel eine Programmiersprache wie BASIC besteht.
2. Mit der "Logik" und dem Aufbau von Programmen, also mit der Methode, durch die eine Idee im Kopf des Programmierers zum Programm im Computer wird – und zwar zu einem Programm, das nicht nur der Computer "versteht", sondern das Menschen anwenden können.

Mit diesem zweiten Punkt beschäftigt sich unser dritter Band der Reihe "Computer kinderleicht". Anhand dieses Buches lernen Kinder, vielseitige, interessante und witzige Programme zu schreiben. Dabei bleibt der hierzu notwendige Vorrat an BASIC-Ausdrücken und Funktionen überschaubar.

Wie bei einer natürlichen Sprache kommt es bei einer "Computersprache" nicht allein auf den Wortschatz an, sondern auch darauf, was man daraus macht, nämlich auf die Einfälle und Ideen. Hier ist unsere Phantasie herausgefordert.

Dies verlangt nicht einfach nur braves Nachdenken oder gar stures Pauken. Es setzt auf die Eigeninitiative seiner Leser. Die Kinder und Jugendlichen lernen spielend, ein Programm schrittweise von der Grundidee über den Plan bis zum fertigen Programm zu entwickeln und auszugestalten.

Weder ballern hier die Kinder mit dem Joystick auf "bewegliche Ziele" noch messen sie sich in einseitigen Reaktions-Wettkämpfen. Statt dessen führen sie unterhaltsame und zuweilen verblüffende Dialoge mit dem Computerprogramm. Wir zeigen, wie sich schon mit einfachen Mitteln, dafür aber mit viel Computerlogik und noch mehr Computerphantasie, amüsante und sogar nützliche Programme austüfteln lassen. Dabei haben wir bemerkt, daß Kinder mit der gleichen Begeisterung Programme entwickeln, mit der sie diese auch anwenden. Und

wir haben festgestellt, daß solche "Anwendungen" wiederum das Lernen selbst in anderen Bereichen schmackhaft machen können.

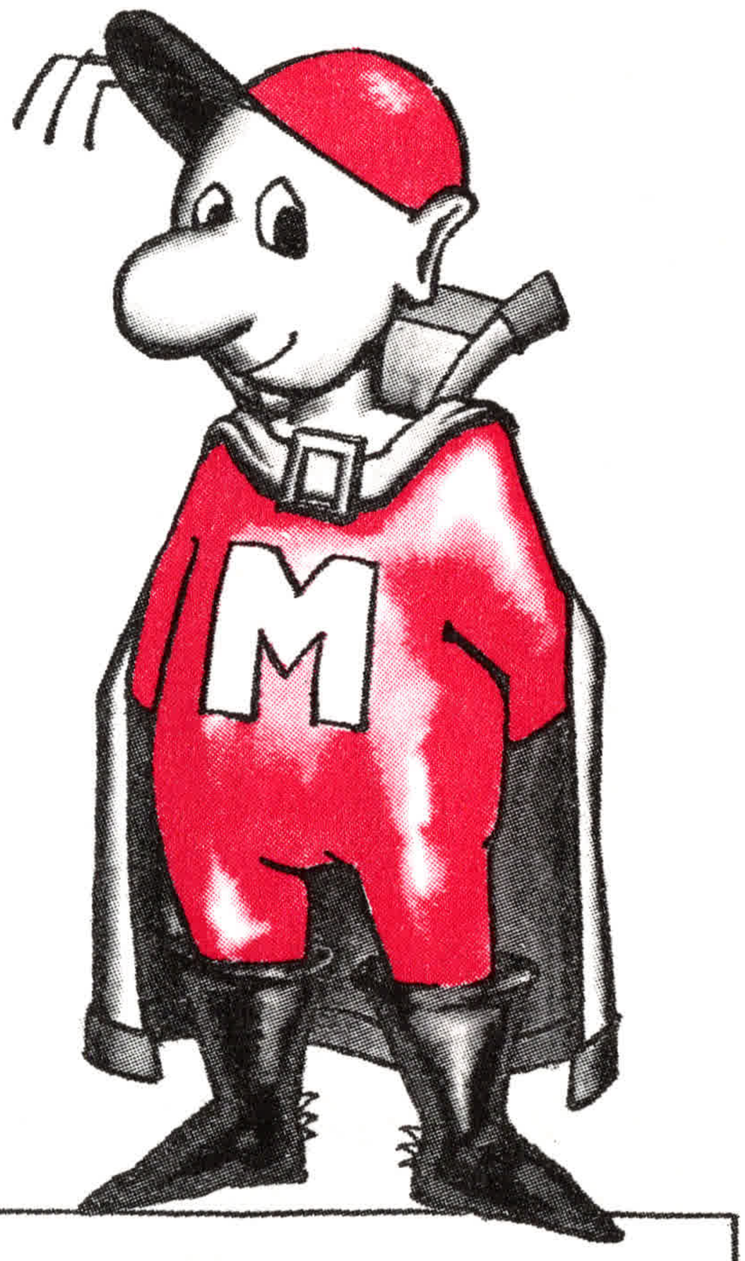
Unsere Reihe geht von dem Gedanken des "sanften Programmierens" aus. Computer sind nicht in erster Linie Feuerleitstände von Raketen oder Spielplätze für Retortenhelden auf der Jagd nach Weltrauminvasoren und Würgemonstern.

Die Hersteller von Spiel- und Lern-Software bieten zwar mittlerweile auch pädagogisch durchaus sinnvolle und empfehlenswerte Produkte an. Doch neben dem Spaß an diesen zweifellos leckeren "Fertiggerichten" liegt uns auch daran, den Spaß am Computer kreativ auszuweiten: Die wahre Computerei ist das "Selberkochen" der eigenen Programme! Erst das eigene Programmieren erschließt uns die Möglichkeiten der "Maschine" umfassend. Damit der Leser trotzdem nicht alles selber machen muß, findet er auch im Anhang dieses Bandes wieder fertige Programmlistings zum Abtippen.

Wir stellen uns den Computer nicht als Spieldroge vor, sondern als eine von vielen Möglichkeiten zur Entfaltung von schöpferischen Fähigkeiten – ein neues und faszinierendes Mittel, um Technik und Phantasie, Lernen und Spaß miteinander zu verbinden.

Wichtig!

Die Programme dieser Reihe wurden auf jener Version des Commodore 64 entwickelt, bei der pro BASIC-Programmzeile maximal 80 Zeichen möglich sind (genau 2 Bildschirm-Zeilen). Es gibt anscheinend C 64-Versionen, bei denen nicht so viele Zeichen in eine BASIC-Programmzeile passen. Dies macht sich beim Abtippen der Listings unangenehm bemerkbar. Vor allem sind offenbar Commodoregeräte betroffen, die vor 1984 hergestellt worden sind. Wahrscheinlich ist dies bei Geräten, die ab 1985 gebaut worden sind, kaum noch zu befürchten.
Also: bitte nicht auf die Autoren schimpfen!



COMPUTER-MAX traut sich alles zu. Das meiste Klappt dann auch.

Wir unterhalten uns mit dem Computer

Wenn Leute miteinander reden, nennen wir das Gespräch zwischen ihnen auch Dialog. Dialoge hören wir jeden Tag – zum Beispiel so einen:

Herr Knependonk: "Guten Tag, Frau Pusemuckel. Wie geht es Ihnen?"

Frau Pusemuckel: "Oh, danke, kann nicht klagen. Und Ihnen?"

Herr Knependonk: "Na ja, nicht schlecht – man lebt. Auf Wiedersehen!"

Frau Pusemuckel: "Wiedersehen, Herr Knependonk; schönen Tag noch!"

Nichts Besonderes. Man begegnet sich und ist freundlich zueinander.

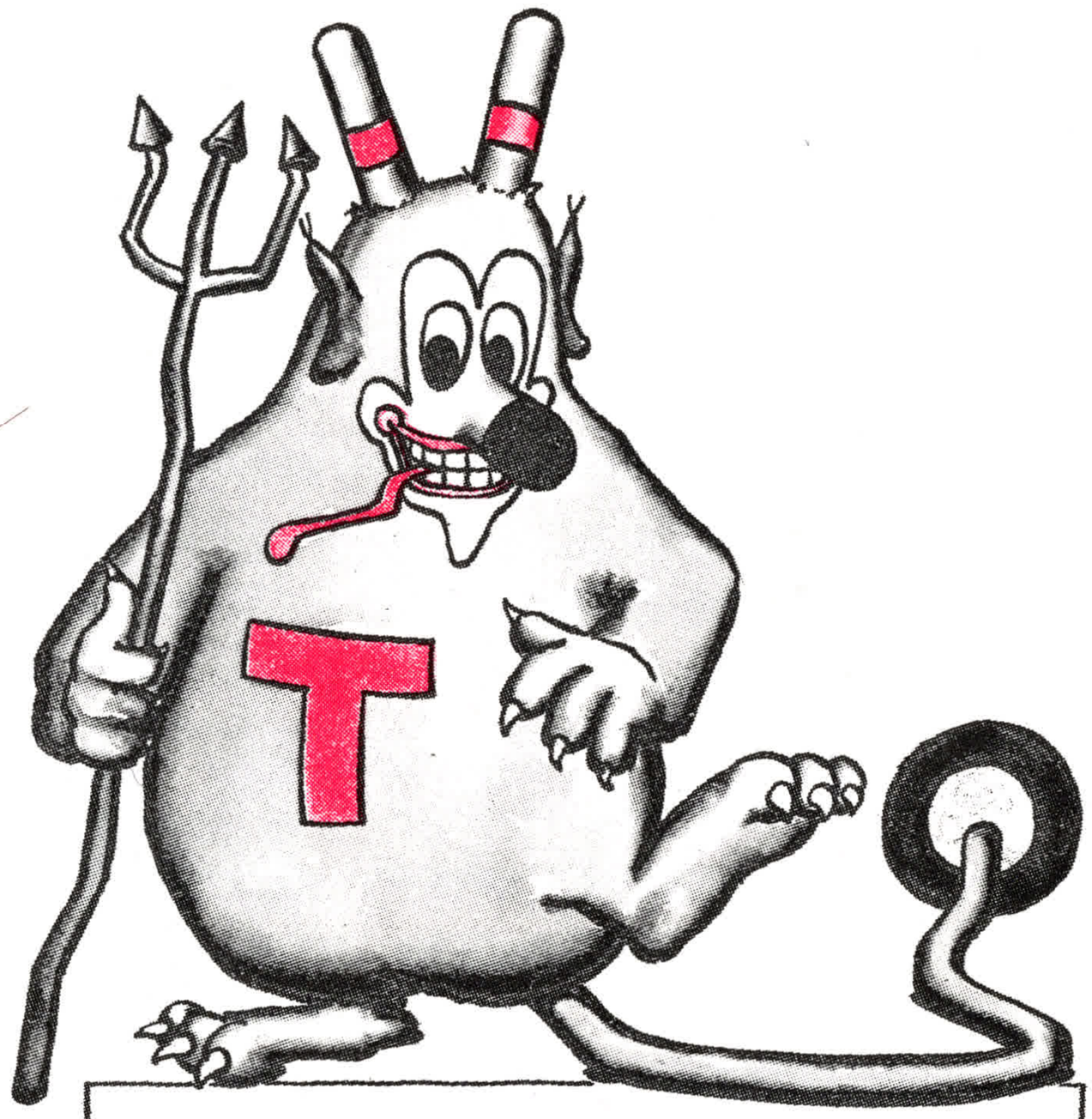
Ein anderer Dialog:

Tourist: "Können Sie mir bitte sagen, wie ich zum Rathaus komme?"

Polizist: "Tja, da gehen Sie am besten zuerst die Hoffmannstraße hinunter, bis Sie an ein großes Kaufhaus kommen – das City-Einkaufszentrum. Da biegen Sie links ab, in die Goethestraße. Von dort immer geradeaus, bis zu einem großen Platz. An dem sehen Sie das Nationaltheater. Und links davon führt die Klopstockstraße direkt zum Rathaus."

Tourist: "Aha, gut. Herzlichen Dank!"

Polizist: "Bitte sehr, gern geschehen!"



Der Teufel THEOBALD hat viel im Kopf, hauptsächlich Unsinn.

Auch nichts Weltbewegendes. Aber immerhin ist so ein Dialog für einen der beiden Gesprächspartner wichtig. Ganz im Gegensatz zum ersten Dialogbeispiel hat hier das Gespräch nämlich Folgen für das Verhalten des Touristen. Er nimmt jetzt den Weg, der ihm vom Polizisten angegeben worden ist und findet (sofern er sich die Beschreibung richtig gemerkt hat) zum Rathaus.

Noch ein Dialog:

Kundin: "Entschuldigen Sie bitte, was kostet denn dieser Hut hier?"

Verkäufer: "Zweihundertneunundvierzigfünfundneunzig, gnädige Frau."

Kundin: "Oh, so viel wollte ich nicht ausgeben. Haben Sie in derselben Art nicht noch etwas Preiswerteres?"

Verkäufer: "Aber gewiß doch. Da könnte ich Ihnen zum Beispiel dieses Modell hier anbieten. Es kostet nur einhundertneunundvierzig Mark."

Kundin: "Hm, ja, ich glaube, der gefällt mir. Doch, den nehme ich. Packen Sie ihn mir bitte ein?"

Verkäufer: "Sehr gerne, gnä' Frau – würden Sie mir bitte zur Kasse folgen?"...

Hast du die Unterschiede zwischen den drei Dialogen bemerkt? Beim ersten Dialog haben die Gesprächspartner einander zwar etwas gefragt, aber es waren keine "richtigen" Fragen. Die beiden kennen sich. Und wenn man einander zufällig begegnet, tauscht man Höflichkeitsfloskeln aus – das ist jedenfalls gemütlicher, als wortlos aneinander vorbeizugehen.

Im dritten Dialog haben zwei Menschen einander ihre Wünsche und Angebote mitgeteilt, bis sie gemeinsam eine Lösung für das Anliegen des einen, nämlich der Kundin, gefunden haben.

Der Plappergei PULLMANN weiß eigentlich alles. Und das merken die anderen auch.

Was sollen wir nun vom folgenden Dialog halten?

Herr Mittagstund: "Entschuldigen Sie bitte, dürfte ich wohl kurz Ihr Fernglas benutzen? Ich würde so gerne das Gipfelkreuz vom Großen Haxenstein anschauen."

Mister Sixoclock: "I beg your pardon. I don't understand you. Do you speak English?"

Herr Mittagstund: "Oje, Englisch. Nein, tut mir leid. Nix verstehn."

Hier ist kein Dialog zustande gekommen, weil die beiden Partner nicht dieselbe Sprache sprechen. Und selbst wenn

beide dieselbe Sprache sprechen, kann es trotzdem vorkommen, daß man aneinander vorbeiredet:

So etwas nennt man einen Nonsens- oder einen Unsinnsdialog. Er kommt zustande, wenn einer dem anderen gar nicht richtig zuhört. In unserem Beispiel hat der Teufel natürlich mit Absicht nicht richtig zugehört. Aber zuweilen geschieht es auch, daß zwei Menschen, die miteinander reden, gar keinen wirklichen Dialog miteinander führen. Jeder spricht von etwas

anderem als der Gesprächspartner, manchmal sogar, ohne es zu merken. Es gibt offenbar sehr verschiedene Arten, einen Dialog zu führen oder eben nicht zu führen.

Auch bei der Arbeit mit dem Computer sollte man wissen, welche Art von Dialog man haben möchte, und vor allem: welche Fehler dabei möglich sind. Einen Dialog können wir nämlich auch mit dem Computer zustande bringen. Vorausgesetzt, wir haben dazu das richtige Programm.

Bis jetzt haben wir dem Computer immer nur eine Aufgabe gestellt und sie von ihm auf eine bestimmte Art lösen lassen. Jetzt soll er uns Fragen stellen können. Dazu brauchen wir ein neues BASIC-Wort. Es heißt

INPUT

und bedeutet soviel wie "Eingabe". Gesprochen wird es ausnahmsweise genauso, wie es geschrieben wird. Eigentlich ist alles, was wir dem Computer eingeben, ein "Input". Man kann etwas mit der Tastatur, also über das Keyboard eingeben. Man kann aber auch etwas direkt vom Massenspeicher wie der Datasette oder Floppy eingeben. Für Grafik gibt es besondere Eingabegeräte, zum Beispiel ein Grafik-Tablett. Auch der Joystick ist ein Eingabegerät. Das merkst du zwar nicht, wenn du damit deinen Pac-Man durchs Labyrinth steuerst. Aber es wird sofort klar, sobald wir mit dem Joystick auf dem Bildschirm zeichnen. Einige Grafikprogramme bieten diese Möglichkeit.

Diese Programme benutzen den Joystick als "Bildschirm-Bleistift". Und so ein Bildschirmstift ist auch der "Light-Pen" (gesprochen "laitpen"), den man auf deutsch "Lichtgriffel" nennt. Mit ihm kann man unmittelbar auf dem Bildschirm zeichnen oder schreiben. In manchen Programmen lassen sich mit dem Lichtgriffel bestimmte Dinge auswählen, die der Computer anbietet. Statt etwas einzutippen, muß man nur mit dem Light-Pen auf eine bestimmte Stelle einer Grafik auf dem Bildschirm zeigen. Dann tut der Computer, was man von ihm erwartet.

Solch ein Angebot nennt man übrigens ein "Menü". Es erinnert nämlich an eine Speisekarte in einem Restaurant, wo wir zwischen verschiedenen Speisen und Getränken auswählen können. Unser Vokabelprogramm auf der Diskette zu diesem Buch wird gleichfalls mit Hilfe einer "Menü-Steuerung" bedient. Das BASIC-Listing dazu steht im Anhang. Dieses Listing veranschaulicht den Programmaufbau. In BASIC geschrieben ist

es jedoch sehr langsam. Für die praktische Arbeit mit dem Vokabelpaukprogramm gibt es auf der Diskette zum Buch eine wesentlich schnellere Version.

Sie ist in einer besonderen Programmiersprache geschrieben. Diese Programmiersprache heißt Assembler. Sie belegt im Computer weniger Speicherplatz. Deshalb läuft das Programm um vieles schneller. Auch die Ladezeiten werden kürzer.

Assembler ist jedoch eine sehr viel schwerere Programmiersprache als BASIC. Daher verwenden wir sie in dieser Reihe lediglich, um den praktischen Nutzen unserer "Software" zu steigern. Ansonsten bleiben wir vorerst bei BASIC.

Wie gelangt der Input in den Computer? Es gibt nicht nur viele verschiedenartige Geräte, über die man den Computer mit Eingaben bzw. Input "füttern" kann. Auch der Input selbst kann unterschiedlich aussehen. Er kann aus Zahlen oder Wörtern bestehen, ebenso auch aus Tönen oder Bildern.

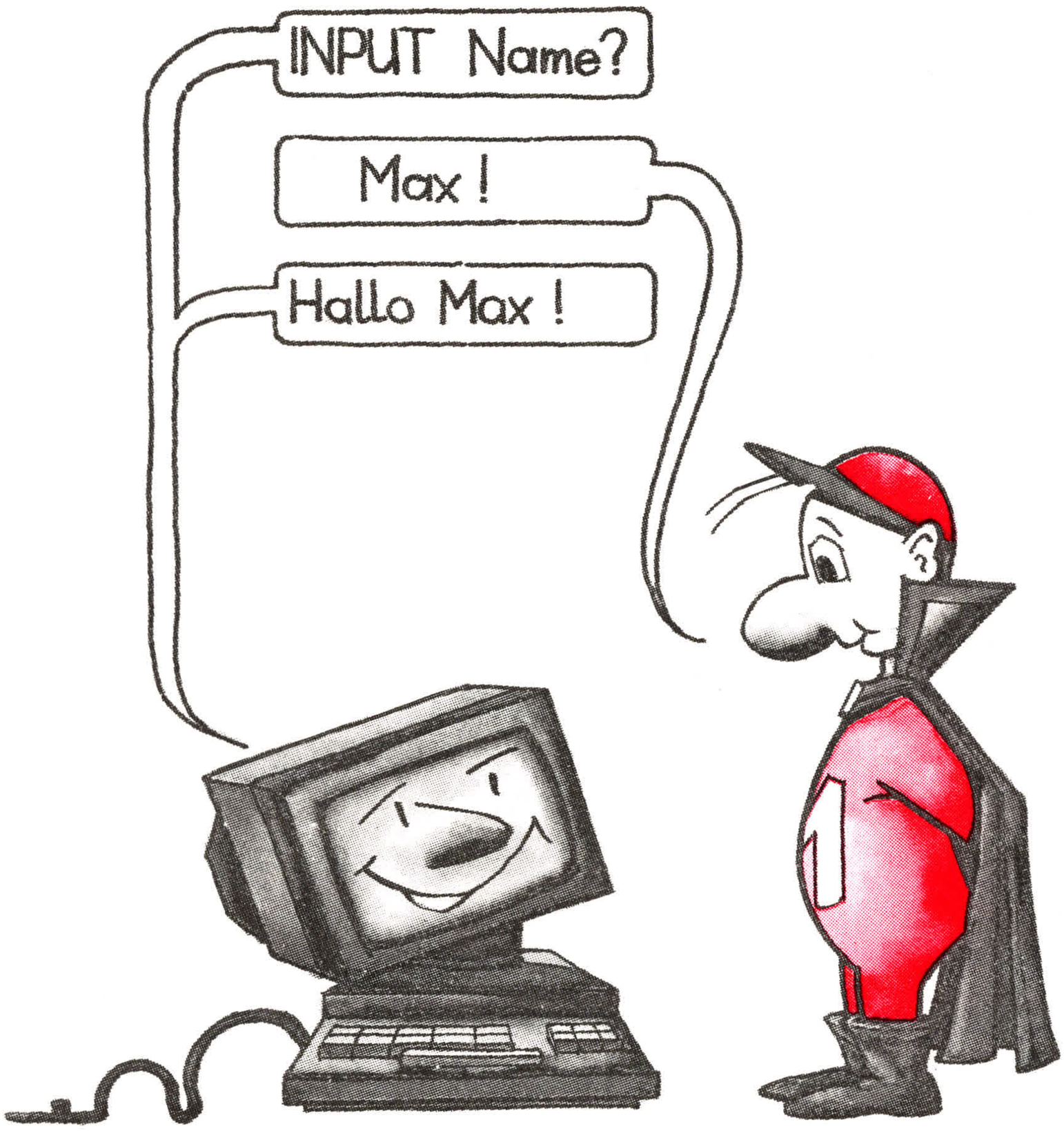
Was der Computer als Lösung oder als Antwort ausgibt, das ist der "Output" (gesprochen "autput"). Auch dieser Output kann ganz verschiedene Formen haben. Er kann auf dem Bildschirm erscheinen, zum Beispiel als Antwort, die aus Zahlen oder Sätzen besteht. Der Output kann auch ein Bild sein oder eine Melodie oder nur ein Geräusch.

Seinen Output kann der Computer nicht nur auf dem Bildschirm, sondern zugleich auch auf dem Drucker ausgeben oder auf einer Diskette speichern. Wenn wir ihn mit Hilfe eines entsprechenden Programms zum "Klavier" machen, kommt der Output zum Beispiel aus einer angeschlossenen Stereoanlage.

Bei unserem Gespräch mit dem Computer wechseln Input und Output, Eingabe und Ausgabe, einander ständig ab. Das spezielle BASIC-Wort INPUT verlangt, daß man an **einer bestimmten Stelle** des Programms eine Eingabe macht. Ohne diese Eingabe kann das Programm nicht weiterlaufen.

Im einfachsten Fall fragt der Computer nach einer Zahl, mit der er rechnen soll. Das erreichen wir durch die Verwendung von Variablen. Im zweiten Band hast du gelernt, daß die Variable ein Zeichen ist, das eine Zahl vertritt. Dieses Zeichen kann ein Buchstabe sein, zum Beispiel A, B, C oder D.

Vielleicht erinnerst du dich auch noch daran, daß wir im zweiten Band von "Computer kinderleicht" ("Lernen, spielen, programmieren") schon mit solchen Variablen ein Einmaleins-Programm geschrieben haben. Wie der INPUT-Befehl verwendet



wird, wollen wir ebenfalls anhand eines kleinen Einmaleins-Programms veranschaulichen.

Wir haben ja schon gezeigt, daß man in einem BASIC-Programm einer Variablen einen bestimmten Wert zuordnen kann. Ein Programm dazu kann so aussehen:

```
10 A=5
20 PRINT A;"*5=";A*5
```

Mit

```
RUN
```

gestartet bringt der Computer als Output folgendes auf dem Bildschirm:

```
RUN
 5 *5 = 25

READY.
```

Abb. HC 1

Denn bei diesem Programm muß ja immer wieder die Zeile 10 geändert werden, sobald du mit einer anderen Zahl rechnen möchtest. Da du mit deiner Cursorsteuerung und den Möglichkeiten zum Radieren noch gut vertraut bist, geht das zwar sicher ziemlich schnell. Aber es gibt eine viel "komfortablere" Lösung für dieses Problem. Denn statt

A=5

(die Variable A **soll** den Wert 5 haben) geben wir ein:

INPUT A

Das Programm fragt jetzt, welchen Wert die Variable A annehmen soll. Das Programm sieht dann so aus:

```
10 INPUT A
20 PRINT A*5
```

In Zeile 10 "fragt" uns also der Computer, mit welcher Zahl wir rechnen wollen. In Zeile 20 rechnet er dann mit der eingegebenen Zahl. Sobald das Programm mit

RUN

gestartet wird, meldet sich der Computer mit einem *Fragezeichen* (links auf dem Bildschirm). Das heißt: Er wartet nun darauf, daß du ihm eine Zahl eintippst, mit der er die vorgeschriebene Rechnung ausführen soll. Du kannst jetzt die Zahl eingeben und mit

RETURN

die Eingabe bestätigen. Der Computer errechnet die Lösung der Aufgabe. Sein Output ist die richtige "Antwort". Das kann so aussehen:

```
10 INPUT A
20 PRINT A*5
RUN
? 4
 20

READY.
```

Abb. HC 2

Beim nächsten Programmstart wird der Unterschied zu unserem ursprünglichen Einmaleins-Programm deutlich. Der

Computer fragt nämlich zuerst wieder nach deiner Eingabe, dem Input. Das Programm läuft dabei nicht immer unverzüglich ab, sondern wartet zunächst auf den jeweiligen Input.

Jetzt kannst du ja mal mit dem neuen BASIC-Befehl

INPUT

ein bißchen herumexperimentieren. Was passiert wohl, wenn du auf das Fragezeichen nach dem INPUT etwas anderes als eine einfache Zahl eintippst (einen Buchstaben zum Beispiel oder einen ganzen Rechenausdruck)? Nach einigen Versuchen sieht es auf deinem Bildschirm wahrscheinlich ungefähr so aus wie in der Abbildung HC 3.

```
10  INPUT  A
20  PRINT  A*5
RUN
?  8
?REDO  FROM  START
?  4-2
?REDO  FROM  START
?  (4-2)
?REDO  FROM  START
?  8/2
?REDO  FROM  START
?  7
 35
READY.
```

Abb. HC 3

Bei Buchstaben und Rechenausdrücken als Input kommt die Fehlermeldung:

?REDO FROM START

Gesprochen wird das "ridu from start". Im Englischen bedeutet es soviel wie "fang noch mal von vorn an" oder "neue Eingabe, bitte". Das Programm akzeptiert für eine Variable als Eingabe nur eine einzige Zahl. Diese Zahl kann aus einer oder mehreren Ziffern bestehen. Etwas anderes akzeptiert der Computer nicht. Bei unserem ursprünglichen kleinen Einmaleins-Programm hatte der Computer für eine Variable, der man keinen Wert zugeordnet hat, automatisch den Wert 0 angenommen. Wenn wir den INPUT-Befehl verwenden, tut er das nicht. Er rechnet auch nicht den Rechenausdruck aus, den wir ihm angeboten haben. Er will eine Zahl, und sonst nichts.

Welchen Grund das hat, werden wir später sehen. Für den INPUT-Befehl kann man nämlich nicht nur nach Zahlen fragen,

sondern auch nach ganz anderen Sachen.

Wenn du unser Mini-INPUT-Programm

```
10 INPUT A
20 PRINT A*5
```

laufen läßt, dann weißt du selbstverständlich, was der Computer mit seinem Fragezeichen verlangt. Im Programm hast du ja die gewünschte Rechenoperation selber eingetragen.

Doch stell dir vor, jemand kennt das Programm nicht und hat es gestartet. Außerdem hat er von Computerei keine Ahnung. Was könnte der mit so einem Fragezeichen anfangen? Sicher, du könntest ihm erklären, daß der Computer damit fragt, welche Zahl mit 5 multipliziert werden soll. Aber diese Erklärung könnte doch ebensogut der Computer selbst liefern. Dazu schreiben wir vor dem

INPUT

noch einige Textzeilen. Sie erklären dem Benutzer, was er tun soll. Das kann zum Beispiel so aussehen wie in Abbildung HC 4.

```
10 PRINT "ICH NEHME DIE ZAHL"
20 PRINT "MAL FUENF."
30 PRINT "DANN KOMMT"
40 PRINT "DAS ERGEBNIS"
50 PRINT A*5
```

Abb. HC 4

Sobald man dieses Programm startet, sieht es auf dem Bildschirm so aus wie in Abbildung HC 5.

```
ICH NEHME DIE ZAHL
MAL FUENF.
DANN KOMMT
DAS ERGEBNIS
? 6
 30

READY.
```

Abb. HC 5

Das Ergebnis, also den Output des Computers, kann man ebenfalls mit "Erklärungen" ergänzen. Diese Ergänzungen

fügen wir als zusätzliche Programmzeilen ein, etwa wie im Beispiel auf Abbildung HC 6.

```
60 PRINT "ICH MEINE DIE ZAHL"
70 PRINT "MACH DEN FRAGEZEICHEN"
80 PRINT "MAL FUENF."
50 INPUT A
60 PRINT A*5
70 PRINT "DIE LOESUNG LAUTET:"
80 PRINT A;"*5=";A*5
```

Abb. HC 6

- Zeile 70 sagt, was der Computer als Output zu bieten hat.
- Zeile 80 zeigt die ganze Rechnung und die Lösung.

Achte bitte genau auf die Trennungszeichen und die Anführungszeichen in Zeile 80.

Zuerst kommt die Variable A. Sie kann nach jedem INPUT mit einem anderen Wert belegt sein. Mit dem Semikolon (;) erreichen wir, daß es *ohne* Zwischenraum oder neue Zeile weitergeht. Der Ausdruck *5 = bleibt hingegen unverändert. Diese sogenannte Rechenoperation soll der Computer genauso ausgeben, wie wir sie einprogrammiert haben. Dann kommt wieder ein Semikolon (;), damit es sofort in derselben Zeile weitergeht. Und A*5 ist die eigentliche Aufgabe, die der Computer zu lösen hat. Genau hinter der Rechenoperation soll schließlich das Ergebnis erscheinen.

Jedesmal, wenn das Programm mit RUN neu gestartet wird, "vergißt" der Computer, welchen Wert man der Variablen A zuvor gegeben hat und fragt nach einem neuen Wert. Das ist dir nach einigen Programmdurchläufen wahrscheinlich schon aufgefallen.

In einem Programm kann INPUT auch öfter als nur einmal vorkommen. Du mußt dich ja nicht unbedingt damit begnügen, daß das Programm im Fünfer-Einmaleins rechnet. Wir können dafür sorgen, daß es uns am Anfang fragt, in welchem Einmaleins gerechnet werden soll. Wir können das Programm fragen lassen, welche Zahl mit welcher malgenommen werden soll. Wie ein solches Programm aussehen kann, siehst du in Abbildung HC 7.

```
10 PRINT "WELCHE ZAHLEN SOLL ICH MALNEHMEN?"
20 INPUT A
30 INPUT B
40 PRINT A;"*";B;"=";A*B
```

Abb. HC 7

Was auf dem Bildschirm zu sehen ist, sobald man das Programm laufen läßt, zeigen wir in Abbildung HC 8.

```
10 PRINT "WELCHE ZAHLEN SOLL ICH MALNEHM
EN?"
20 INPUT A
30 INPUT B
40 PRINT A;"*";B;"=";A*B
RUN
WELCHE ZAHLEN SOLL ICH MALNEHMEN?
? 4
? 6
 4 * 6 = 24
READY.
```

Abb. HC 8

Dieses Programm läßt sich noch etwas verfeinern. Stell dir vor, wir wollen zu Programmbeginn das gewünschte Einmaleins aussuchen.

Sobald das Programm gestartet ist, soll also der erste INPUT nur einmal, der zweite jedoch stets von neuem eingegeben werden. Das heißt: Das Programm soll die Zeile 20 nur einmal abarbeiten, die Zeile 30 aber immer wieder. Dazu schicken wir unser Programm immer wieder zurück in die Zeile 30, sobald es eine Aufgabe gelöst hat.

Unser BASIC-Wort dafür ist – wie du vielleicht noch weißt –

GOTO

Vielleicht versuchst du selbst erst, das Programm entsprechend umzuschreiben, bevor du dir unseren Vorschlag ansiehst.

Das Programm könnte so aussehen wie in Abbildung HC 9.

Abb. HC 9

```
10 PRINT "WELCHE ZAHLEN SOLL ICH MALNEHMEN?"
20 INPUT A
30 INPUT B
40 PRINT A;"*";B;"=";A*B
50 PRINT "NEUE ZAHL MAL";A:GOTO 30
```

Wie es abläuft, siehst du in Abbildung HC 10.

Um zu demselben Ergebnis zu kommen, hätte man dieses Programm auch anders aufbauen können, etwa indem wir mehr Zeilen verwenden.

Wir haben in Zeile 50 einen Hinweis eingefügt, damit der Benutzer des Programmes weiß, was er als nächstes tun muß.

```
10 PRINT "WELCHE ZAHLEN SOLL ICH MALNEHM
EN?"
20 INPUT A
30 INPUT B
40 PRINT A;"*";B;"=";A*B
50 PRINT "NEUE ZAHL MAL";A:GOTO 30
RUN
WELCHE ZAHLEN SOLL ICH MALNEHMEN?
? 3
? 4
 3 * 4 = 12
NEUE ZAHL MAL 3
? 5
 3 * 5 = 15
NEUE ZAHL MAL 3
? 8
 3 * 8 = 24
NEUE ZAHL MAL 3
?
 3 * 8 = 24
NEUE ZAHL MAL 3
```

Abb. HC 10

Auch dieser Hinweis gehört zum Dialog mit dem Computer. Du kannst ihn nach deinen eigenen Vorstellungen gestalten.

Noch etwas fällt auf, wenn man dieses Programm laufen läßt: Sobald ein Programm mit RUN neu gestartet wird, "vergißt" es den Wert all jener Variablen, die beim vorherigen Durchlauf angegeben worden sind. Es fragt den Benutzer nach einem neuen Wert. Es vergißt den Wert aber auch dann, wenn es an eine INPUT-Zeile zurückkehrt.

Gegenwärtig verhält sich unser Einmaleins wie ein Perpetuum mobile. Von selber hält es nicht an. Es kehrt zwar immer wieder zur Zeile 30 zurück. Beenden läßt es sich hingegen nur mit Hilfe der Taste

RUN STOP

Aus dieser "unendlichen Geschichte" können wir jetzt ein Programm machen, das nach einer bestimmten Anzahl von Durchläufen beendet wird. Hierfür ließe sich beispielsweise eine "Rundenzählmaschine" einbauen. Die kennst du vielleicht noch aus dem Band 2 dieser Reihe ("Lernen, spielen, programmieren").

Dazu belegst du eine Variable mit einem bestimmten Anfangswert. In unserem Fall nehmen wir als Wert am besten die Eins. In Zeile 1 schreiben wir demnach

```
K=1
```

Es könnte ebenso auch die Zeile 2 oder 5 sein. Jedenfalls ist der Wert in dieser vorangestellten Programmzeile bestens aufgehoben. Hauptsache, die Zeilennummer ist niedriger als 10. Die Variable soll ja nur für den Anfang des Programmes gelten.

Hat das Programm die erste Runde hinter sich, soll der K-Wert um 1 erhöht werden usw. Folglich kommt der Ausdruck

K=K+1

in Zeile 45. An dieser Stelle ist die erste Aufgabe gelöst, und der Computer erwartet die nächste. Und dann soll das Programm überprüfen, ob K einen bestimmten Wert erreicht hat. Nehmen wir als Beispiel 20. Sobald es 20 als diesen K-Wert vorfindet, soll das Programm anhalten.

Unser BASIC-Wort dafür ist

END

Die eingefügte Zeile kann so aussehen:

```
45 K=K+1:IF K=21 THEN END
```

Die zwanzigste Runde wird also noch gedreht. Zur einundzwanzigsten Runde tritt unser Programm nicht mehr an.

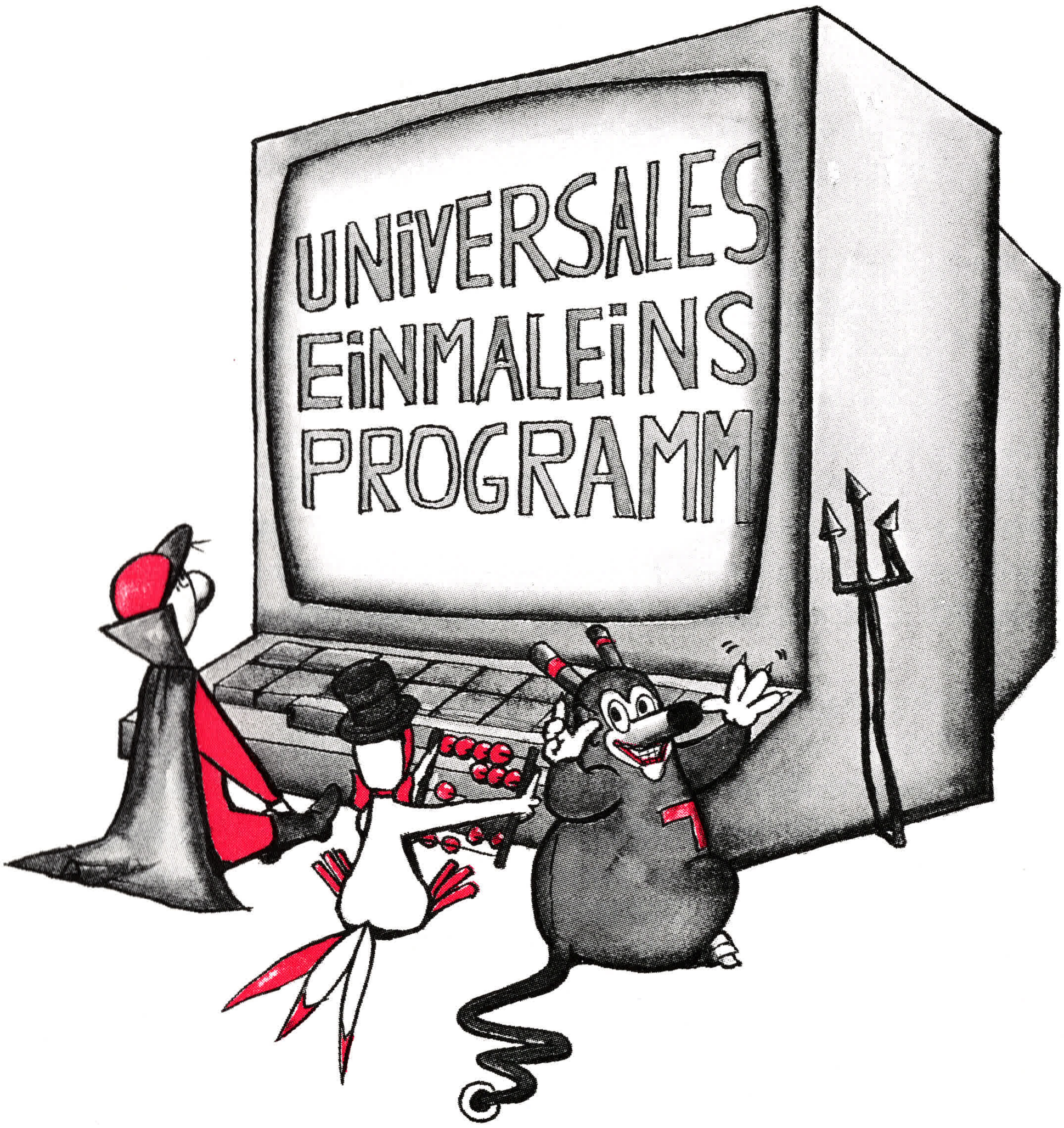
Das ganze Programm sieht bis jetzt so aus wie in Abbildung HC 11.

```
1  K=1
10  PRINT "WELCHE ZAHLEN SOLL ICH MALNEHMEN?"
20  INPUT A
30  INPUT B
40  PRINT A;"*";B;"=";A*B
45  K=K+1:  IF  K=21  THEN  END
50  PRINT "NEUE ZAHL MAL";A:GOTO 30
```

Abb. HC 11

Falls deine Fassung äußerlich von unserem Beispiel geringfügig abweicht, weil du die eine oder andere Programmzeile etwas anders geschrieben hast, macht das nichts. Hauptsache das Programm läuft genausogut.

Mit diesem Programm kann man zwar schön arbeiten, aber dazu müßte man eigentlich immer genau wissen, wie es aufgebaut ist. Falls nicht, ist der Benutzer wahrscheinlich ziemlich überrascht, daß sich das Programm auf einmal kommentarlos



"verabschiedet", nachdem man eine Weile damit gerechnet hat. Bauen wir also wieder eine Textzeile ein, die dem Benutzer genauer sagt, was ihn erwartet – vielleicht so:

```
5 PRINT "ICH MACHE   20 RECHNUNGEN."
```

Wer es besonders bequem haben will, kann auch noch einen zusätzlichen Programmteil einbauen. Dieser sagt dem Benutzer, wie viele Rechenoperationen er den Computer bereits hat lösen lassen und wie viele Berechnungen noch kommen.

Dazu bauen wir die Zähl-Variable K in einen Satz ein. Das läßt sich mit Hilfe der folgenden Programmzeilen erledigen:

```
41 PRINT "ICH HABE BISHER ";K;" MAL GERECHNET."
42 PRINT "ICH RECHNE NOCH ";21-K;" MAL."
```

So sieht das fertige und bereits recht benutzerfreundliche Einmaleins-Programm aus. Damit können wir den Computer alle Multiplikationen ausführen lassen, auch die schwierigsten. Das Programmlisting findest du in Abbildung HC 12.

```
1 K=1
5 PRINT "ICH MACHE 20 RECHNUNGEN."
10 PRINT "WELCHE ZAHLEN SOLL ICH MALNEHMEN"
20 INPUT A
30 INPUT B
40 PRINT A;" * ";B;" = ";A*B
41 PRINT "ICH HABE BISHER ";K;" MAL GERECHNET."
42 PRINT "ICH RECHNE NOCH ";20-K;" MAL."
45 K=K+1:IF K=21 THEN END
50 PRINT "NEUE ZAHL MAL ";A:GOTO 30

READY.
```

Abb. HC 12

Fassen wir noch einmal zusammen. Was ist an diesem Programm anders als bei allen vorherigen kleinen und größeren Programmen?

1. Mit INPUT kann der Computer eine Eingabe verlangen. Er erhält auf diese Weise für das Programm immer neue Informationen.
2. Der Computer kann sein Programm selbst erklären. So etwas nennt man auch "Benutzerführung". Dabei sagt er uns sowohl, welche Eingaben er als Input benötigt, als auch, welche Art von Output er wieder ausgibt.

Der Benutzer dieses anwenderfreundlichen Programmes führt mit dem Computer einen Dialog. Mit einem Programm wie dem, das wir gerade geschrieben haben, kann wirklich jeder Anwender arbeiten. Auch ein Benutzer, der von BASIC keine Ahnung hat, ja sogar jemand, der zum erstenmal vor der Computer-Tastatur sitzt.

Die Texte, mit denen sich ein Programm selbst erklärt, können ganz unterschiedlich sein. Sie können einmal knapp und sachlich ausfallen, das andere Mal ein bißchen lustig und sogar albern, sofern dir danach zumute ist.

Jeder kann sein Programm in seinem ganz persönlichen Stil formulieren. So wie man ja auch beim Briefe- oder Aufsatzschreiben seinen eigenen Stil erkennen läßt. Allerdings gibt es dabei einen kleinen Haken: Der Computer rechnet zwar schnel-

ler und genauer als jeder menschliche Rechenkünstler. Doch nachprüfen, ob man im Text einen Rechtschreib- oder Satzfehler gemacht hat, kann der Commodore 64 genauso wenig wie jeder andere Heim- oder Personalcomputer. Dazu bräuchte er ein besonderes, sehr aufwendiges Programm, und hierfür fehlt Homecomputern meist der erforderliche Speicherplatz. Da bleibt nichts anderes übrig, als selbst aufzupassen. Es ist dem Computer auch völlig egal, ob der Anwender den Text begreift oder nicht. Dafür muß allein der Programmierer sorgen.

Genau wie andere Teile eines Programmes kann man auch diese Texte immer wieder ändern und verbessern. Oft merkt man ja erst beim Ausprobieren, daß hier oder da etwas noch nicht genügend klar ausgedrückt ist. Am besten ist, man testet alle Programme ein paarmal mit Freundinnen und Freunden durch.

Ein besonderes Vergnügen kann es sein, einige verblüffende kleine Gags einzubauen. Sehr wirkungsvoll sind solche, die dem Benutzer das Gefühl geben, der Computer sei imstande, "mitzudenken" oder vorwitzige Bemerkungen zu machen, wie ein frecher Schüler. Dann macht die Handhabung eines derartigen Programms noch mehr Spaß.

Will sich zum Beispiel jemand vom Computer unbedingt ausrechnen lassen, wieviel eine Zahl mal 1 oder mal 2 ist, könnte man dazu einen Kommentar ins Programm schreiben. Die eingefügte Zeile kann etwa so aussehen:

```
31 IF B<3 THEN PRINT "NA, FUER SOWAS BRAUCHST DU EINEN
COMPUTER?"
```

Der Computer gibt seinen Kommentar nur, falls der vom Benutzer eingegebene Wert kleiner als drei ist.

Damit sind wir dem Wesen eines guten "Dialog"-Programmes schon etwas nähergekommen:

Sobald der Computer einen Input erhalten hat, soll das Programm darauf reagieren. Zum Beispiel, indem es erst einmal "sortiert" und dann etwas tut, das davon abhängt, wie der Input beschaffen ist. Ist die eingegebene Zahl kleiner als ein bestimmter Wert, soll etwas geschehen, zum Beispiel soll der Computer auf dem Bildschirm eine Bemerkung ausgeben.

Eine andere Möglichkeit: Falls der Input eine bestimmte Zahl ist, dann soll etwas anderes passieren, zum Beispiel soll das

Programm wieder an eine bestimmte Stelle zurückkehren. Unsere Zeile 31 könnte daher auch folgendermaßen aussehen:

```
31 IF B=1 OR B=2 THEN PRINT "ZU LEICHT!":GOTO 30
```

Besonders "gemein" ist es, wenn man dafür sorgt, daß Computer sich bei bestimmten Eingaben "verabschieden":

```
31 IF B<3 THEN PRINT "DAS KANNST DU DOCH
   SELBER RECHNEN! TSCHUESS!":END
```

Das END in dieser Zeile gilt, wie du siehst, nur für den Fall, daß B kleiner als 3 ist – auch wenn es hinter einem Doppelpunkt steht.

Programme, die Eingaben und Ausgaben verwenden, schreibt man meist nach folgendem Schema:

1. Eingabe	(INPUT)
2. Sortieren durch bedingte Verzweigungen	(IF . . .THEN, AND, OR)
3. Bearbeitung oder Output	(PRINT, GOTO, END)

In Abbildung HC 13 siehst du ein Programm, das nach diesem Schema aufgebaut ist. Einige unserer Gags sind darin bereits enthalten. Schau es dir noch einmal an. Vielleicht hast du Lust, einige zusätzliche Bedingungen oder Kommentare einzubauen.

Beispielsweise ließe sich durch eine IF . . . THEN-Verzweigung verhindern, daß der rücksichtslose Anwender den armen Computer neckt und mit allzu großen Zahlen füttert. Schließlich handelt es sich ja um ein Programm zum Üben. Doch hierfür könnte man ja folgende Zeile ins Programm häkeln:

```
31 IF B>1000 THEN PRINT "DAS IST MIR ZU
   SCHWER! ICH MUSS MICH ERHOLEN.
   TSCHUESS!":END
```

Fast könnte man sagen, daß du den Computer mit einer derartigen Programmzeile regelrecht dazu bringst, sich zu "verstellen". Denn er ist ja imstande, mühelos noch viel größere

```
1 K=1
5 PRINT"␣ICH MACHE 20 RECHNUNGEN."
10 PRINT "WELCHE ZAHLEN SOLL ICH MALNEHMEN"
20 INPUT A
30 INPUT B
31 IF B<3 THEN PRINT "DAS KANNST DU DOCH SELBER RECHNEN. TSCHUESS!":END
40 PRINT A;" * ";B;" = ";A*B
41 PRINT "ICH HABE BISHER ";K;" MAL GERECHNET."
42 PRINT "ICH RECHNE NOCH ";20-K;" MAL."
45 K=K+1:IF K=21 THEN END
50 PRINT "NEUE ZAHL MAL ";A:GOTO 30
```

Abb. HC 13

```
1 K=1
5 PRINT"␣ICH MACHE 20 RECHNUNGEN."
10 PRINT "WELCHE ZAHLEN SOLL ICH MALNEHMEN"
20 INPUT A
30 INPUT B
31 IF B<3 THEN PRINT "DAS KANNST DU DOCH SELBER AUSRECHNEN.   TSCHUESS!":END
32 IF B>1000 THEN PRINT "DAS IST MIR ZU SCHWER!MUSS MICH ERHOLEN!TSCHUESS!":END
40 PRINT A;" * ";B;" = ";A*B
41 PRINT "ICH HABE BISHER ";K;" MAL GERECHNET."
42 PRINT "ICH RECHNE NOCH ";20-K;" MAL."
45 K=K+1:IF K=21 THEN END
50 PRINT "NEUE ZAHL MAL ";A:GOTO 30

READY.
```

Abb. HC 14

Zahlen zu berechnen. Außerdem muß sich ein Computer, wie wir wissen, sowieso nie "erholen".

Besonders reizvoll ist es, den Dialog zwischen Computer und Benutzer so zu programmieren, daß der Computer dem Benutzer fast ein bißchen menschlich vorkommt. Dazu gehört, daß er dem Anwender immer sehr höfliche Antworten gibt. Dazu gehört aber auch, daß er sich keine "Unverschämtheiten" gefallen läßt. In Abbildung HC 14 siehst du ein Programm, mit dem du vor allem solche Leute verblüffen wirst, die noch nicht wissen, wie ein Programm funktioniert. Selbstverständlich könntest du die Texte auch ganz anders formulieren. Diese Version ist ja nur ein Vorschlag.

Wie das Programm abläuft, zeigen wir in Abbildung HC 15 und HC 16.

```
ICH MACHE 20 RECHNUNGEN.
WELCHE ZAHLEN SOLL ICH MALNEHMEN
? 2
? 6
 2  *  6  =  12
ICH HABE BISHER  1  MAL GERECHNET.
ICH RECHNE NOCH  19  MAL.
NEUE ZAHL MAL  2
? 7
 2  *  7  =  14
ICH HABE BISHER  2  MAL GERECHNET.
ICH RECHNE NOCH  18  MAL.
NEUE ZAHL MAL  2
? 1
DAS KANNST DU DOCH SELBER AUSRECHNEN.
TSCHUESS!

READY.
```

Abb. HC 15

```
ICH MACHE 20 RECHNUNGEN.
WELCHE ZAHLEN SOLL ICH MALNEHMEN
? 23
? 3
 23  *  3  =  69
ICH HABE BISHER  1  MAL GERECHNET.
ICH RECHNE NOCH  19  MAL.
NEUE ZAHL MAL  23
? 4
 23  *  4  =  92
ICH HABE BISHER  2  MAL GERECHNET.
ICH RECHNE NOCH  18  MAL.
NEUE ZAHL MAL  23
? 1222
DAS IST MIR ZU SCHWER!MUSS MICH ERHOLEN!
TSCHUESS!

READY.
```

Abb. HC 16

Etwas stört uns noch an diesem Programm. Es sind die vielen "krummen" Zeilenzahlen. Zwar beeinträchtigen sie nicht den reibungslosen Programmablauf, aber an manchen Stellen hätte man es schon schwer, noch weitere Programmzeilen einzufügen (falls uns noch was Wichtiges einfiele).

Außerdem erleichtert es den "Durchblick", wenn das Listing in übersichtlichen Zehnerschritten angelegt ist. Zwar ließe sich das Programm ohne weiteres abändern, indem wir durch umständliches "Radieren" die unerwünschten Zeilennummern löschen und überschreiben.

Aber das ist sehr kompliziert. Zum Beispiel müßten wir dann auch die Zahlen hinter dem GOTO verändern. Selbst dem erfahrenen Programmierer unterlaufen dabei immer wieder Fehler. Und schließlich kann es soweit kommen, daß ein Programm nicht mehr funktioniert, an dem man mit so viel Mühe hingepfriemelt hat.

Im BASIC vieler Heimcomputer gibt es daher den Befehl

RENUM

Das ist die Abkürzung von "RENUMBER" (gesprochen "rinamber") und bedeutet "neu numerieren". Das BASIC des C 64 verfügt über diesen Befehl leider nicht.

Es gibt auf dem Markt allerdings einige BASIC-Erweiterungen für unseren 64er. Bei diesen Erweiterungen hat man den RENUM-Befehl nachträglich hinzugefügt. Doch die BASIC-Erweiterungen hat sich ja noch nicht jeder zugelegt. Außerdem muß man dabei oft noch eine Reihe zusätzlicher neuer Ausdrücke hinzulernen.

Wir haben uns deshalb etwas anderes ausgedacht: Das erste Programm auf der Diskette zu diesem Buch heißt "Neue Nummern". Damit läßt sich der Befehl

RENUM

auch auf dem C 64 ausführen.

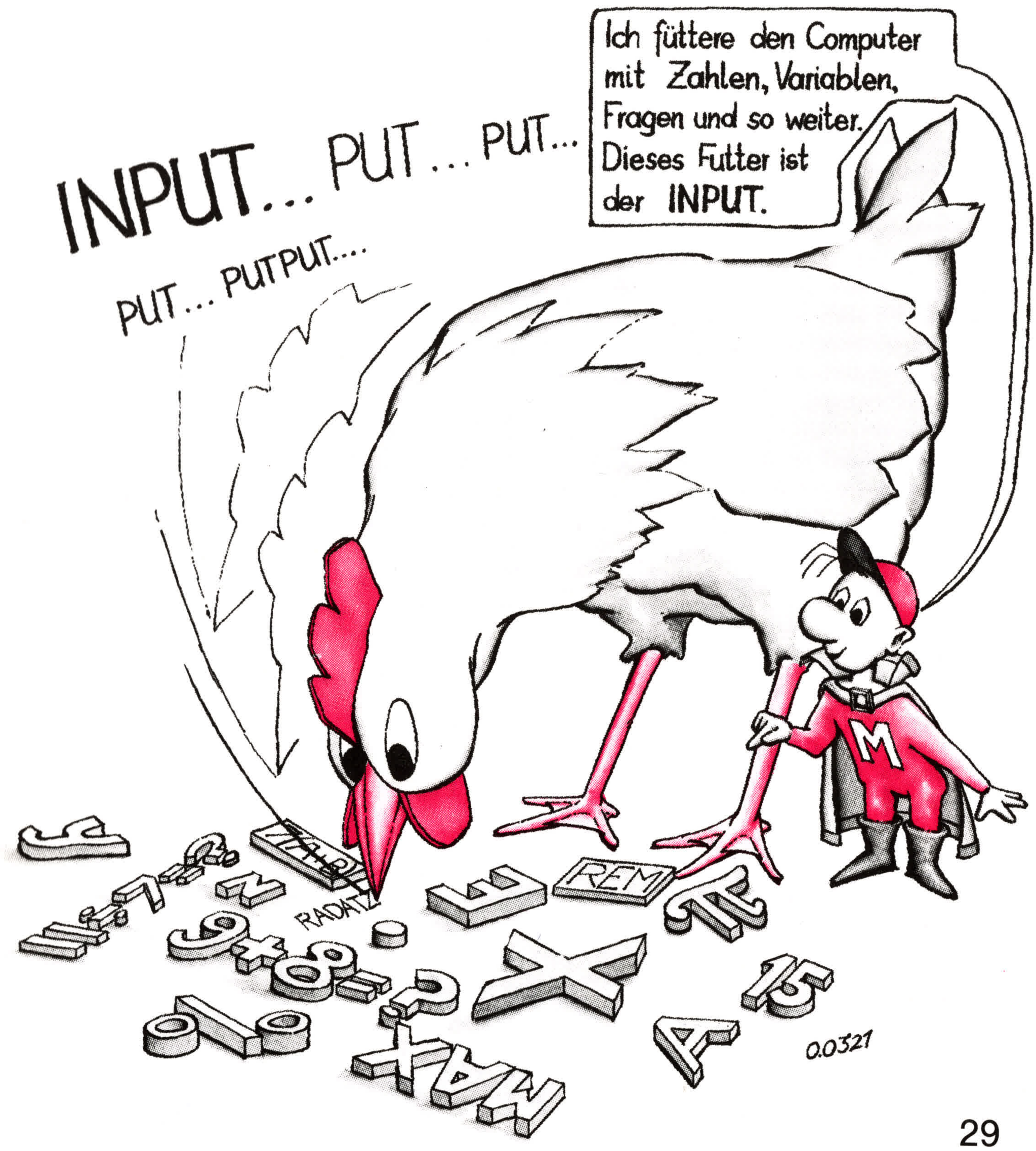
Allerdings muß man dieses "Dienst"-Programm jedesmal neu laden. Am besten ist es, du lädst es immer, bevor du etwas längere Programme schreiben möchtest. Aber denk dran: Wenn sich im Computer bereits ein geladenes Programm befindet, dann geht dieses verloren, sobald ein neues geladen wird. Das Einmaleins-Programm etwa, das wir in diesem Kapitel entwickelt haben, würde somit aus dem Speicher gelöscht, sobald

du nun das Programm "Renumber" laden würdest. Speichere das Einmaleins-Programm daher unbedingt vorher auf Diskette oder Kassette.

Nachdem "Renumber" geladen ist, kannst du auch dein eigenes Programm wieder laden. Dann stören die beiden Programme einander überhaupt nicht. Der Trick dabei ist nämlich, daß "Renumber" mit sehr hohen Zeilennummern beginnt. Auch "Renumber" haben wir zu einem "selbsterklärenden Programm" ausgebaut. Das Listing dazu und einige Erklärungen findest du im Anhang.

INPUT oder wie schütze ich mein Programm vor unbekömmlichem Datenfutter?

Input ist eigentlich alles, was wir dem Computer "mitteilen", also eingeben. Diese Eingabe kann entweder in seiner jeweiligen Sprache erfolgen, und zwar über die Tastatur, oder auf andere Weise, zum Beispiel durch den Joystick. Input ist alles, womit man den Computer "füttert". Output hingegen ist die Antwort, die der Computer "ausgibt". Der Output kann ebenfalls verschiedene Formen haben: die Lösung einer Aufgabe zum Beispiel, eine Bildschirmzeichnung, Töne und vieles mehr.

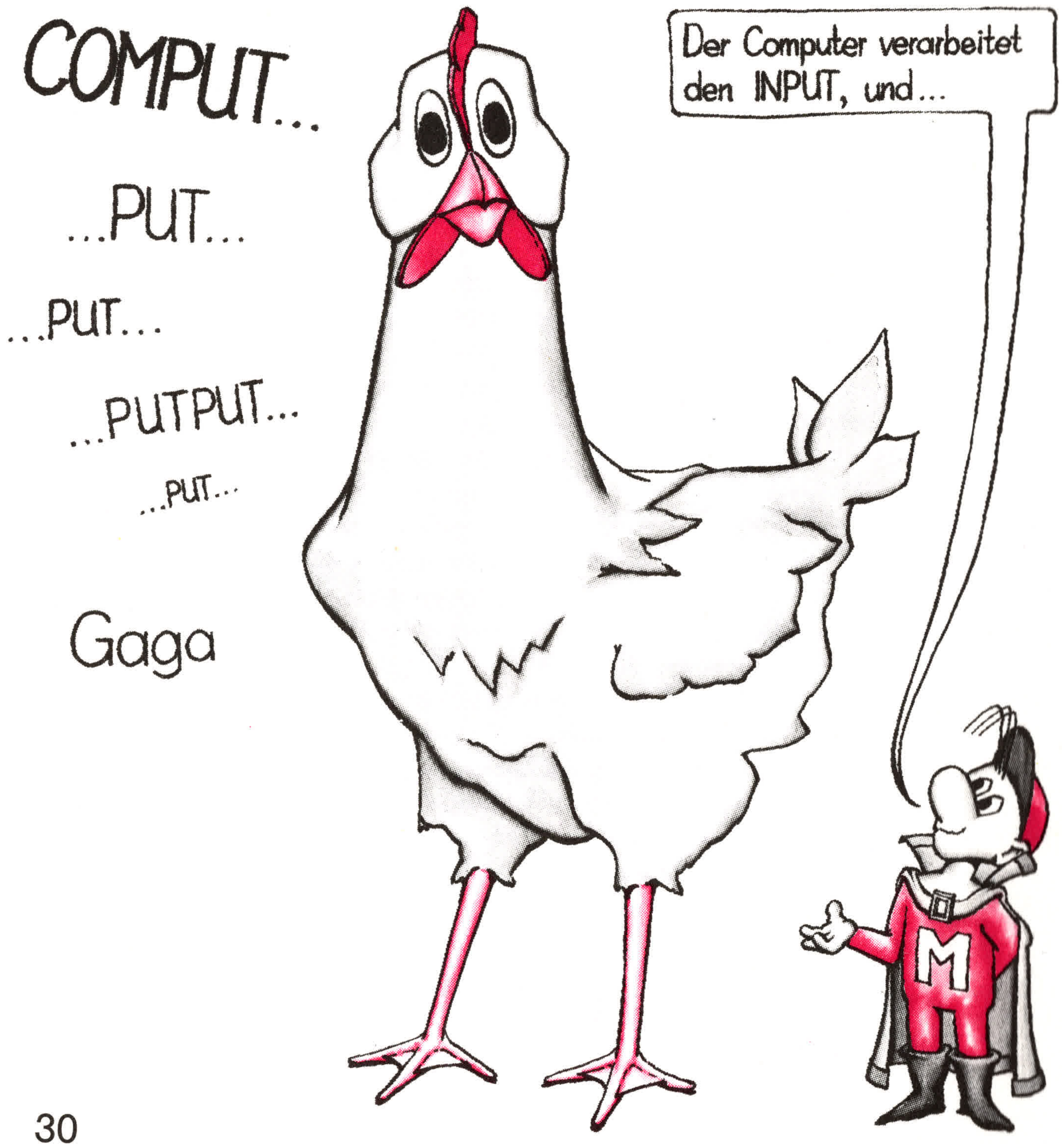


Zwischen Input und Output, also zwischen Eingabe und Ausgabe, geschieht im Computer das Wesentliche, nämlich die "Verarbeitung". Wie der Computer den Input verarbeitet, hängt vom Programm ab. Ohne Programm gibt es keine Ausgabe, sofern man einmal davon absieht, daß der Computer ein paar eingebaute Programme hat, wie etwa die Grundrechenarten.

Jeder Computer arbeitet in der Reihenfolge

Eingabe
Verarbeitung
Ausgabe

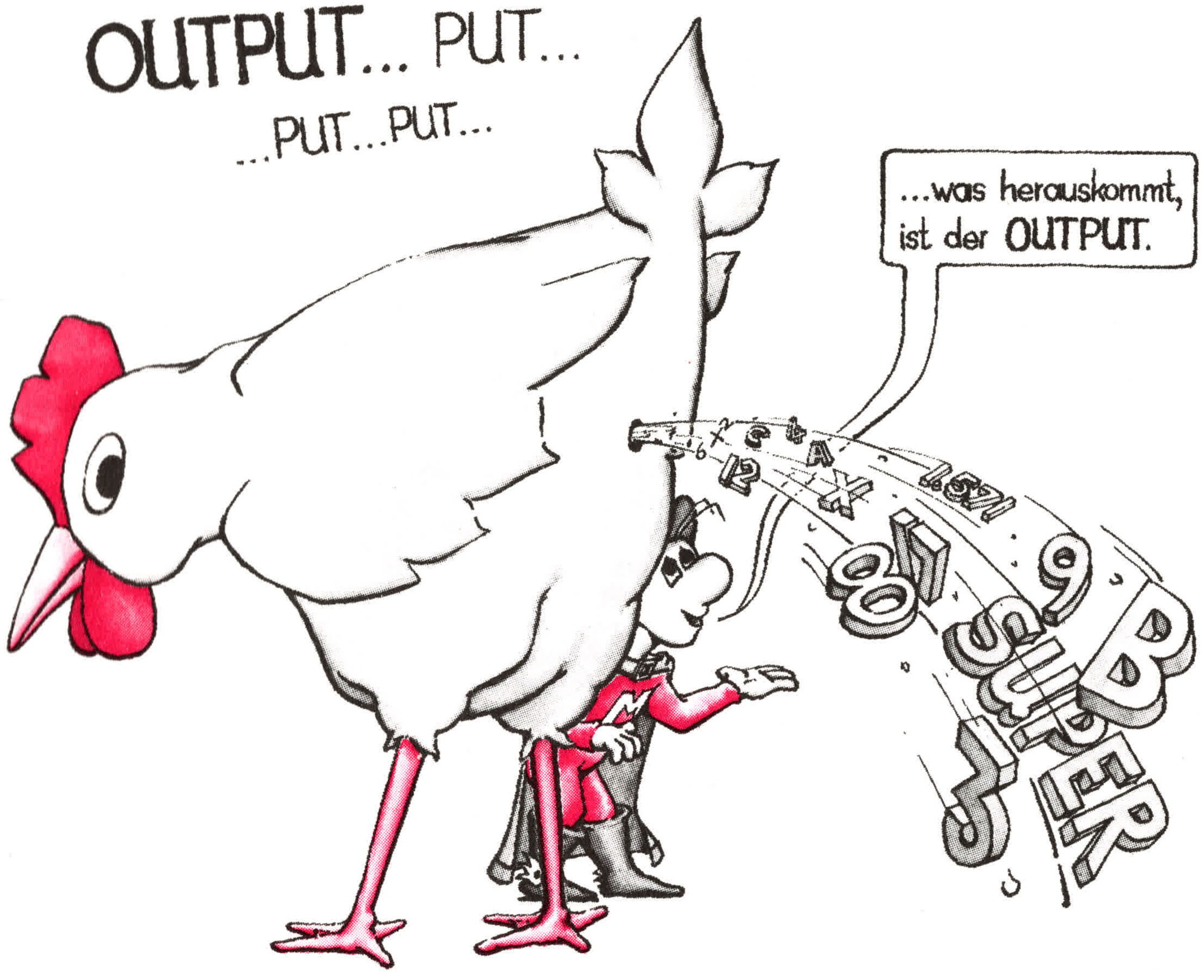
Die Anfangsbuchstaben der drei Wörter ergeben den Ausdruck EVA. Deshalb bezeichnen wir diese Arbeitsweise des Computers auch als das EVA-Prinzip. Das klingt sicher nicht übermäßig geistreich, aber man kann es sich gut merken.



Falls im Programmablauf etwas nicht klappt, gilt es zunächst festzustellen, bei welcher der drei Stufen der Fehler liegen könnte.

1. Liegt der Fehler bei der Eingabe? Haben der Computer oder das Programm etwas nicht oder falsch verstanden? Ist vielleicht ein falsches Peripheriegerät angeschlossen? (Tastatur, Bildschirm, Diskettenlaufwerk, Drucker). Versuche einmal, etwas Sinnvolles einzugeben, wenn gerade der Joystick angeschlossen und auf Dauerfeuer eingestellt ist – dann merkst du rasch, was wir meinen.
2. Liegt der Fehler bei der Verarbeitung? Enthält das Programm einen Fehler? Ist der Rechner selbst nicht in Ordnung?
3. Oder liegt der Fehler bei der Ausgabe? Ist das Ausgabegerät nicht das richtige oder ist es falsch angeschlossen? Läßt sich etwas verändern – zum Beispiel Lautstärke, Farbeinstellung, Drucker-Parameter?

Liefert der Computer nicht den gewünschten Output, so kann das die verschiedensten Ursachen haben. Deshalb sollte man sich gemäß dem EVA-Prinzip zunächst überlegen, an welcher Stelle etwa der Fehler aufgetreten sein könnte.



Einen OUTPUT-Befehl als Gegenstück zum Befehl INPUT gibt es in BASIC nicht. Andererseits sind die Befehle

PRINT

und

RUN

ohnehin bereits Aufforderungen an den Computer, etwas "out-zuputten", also etwas auszugeben.

Für Leute, die täglich mit Computern oder Robotern zu tun haben, gehören die Begriffe Input und Output zum alltäglichen Wortschatz. Diese Ausdrücke werden aber auch dort verwendet, wo gar keine Computer im Spiel sind. Zum Beispiel sagt man: "Da fehlt mir noch der richtige Input", wenn man ausdrükken will, daß man bei einem bestimmten Problem noch zu wenig Informationen hat. Oder man sagt: "Da hatte ich gerade den falschen Input", wenn man erklären will, daß man danebengeraten hat, weil man nicht die richtigen Informationen zur Hand hatte. Ebenso könntest du sagen: "Ich habe noch meinen Output zu erledigen", falls du noch deine Hausaufgaben machen mußt.

Daß Menschen sich mit Maschinen vergleichen, ist nichts Neues. Doch sollten wir uns vergegenwärtigen, daß man sich dabei entweder dümmer macht, als man ist, oder sich sehr unbarmherzige Maßstäbe anlegt – Menschen werden müde und haben das Recht dazu, Maschinen kaum. Menschen kann man auch nicht programmieren. Und wenn sie mal etwas Unvorhergesehenes tun, haben sie auch keinen "Programmfehler". Nur Maschinen sind (weitgehend) berechenbar.

Auch dein Computer ist "nur" eine Maschine. Daran ändert sich auch dann nichts, wenn du anhand deiner Programmierkünste den Eindruck hervorrufen kannst, der Computer sei imstande, selbständig zu denken oder selbst zu "entscheiden".

Das läßt sich vor allem mit Hilfe des INPUT-Befehls besonders gut ermöglichen. Durch den Input hat der Computer nämlich Gelegenheit, sozusagen etwas über denjenigen zu "erfahren", der das Programm benutzt. Er kann diese Informationen weiterverarbeiten, bis sie als Output wieder beim Benutzer ankommen.

Warum nicht zum Beispiel als kleines Kompliment?

Zunächst lassen wir den Computer in einem kleinen Programm das Alter des Benutzers errechnen:

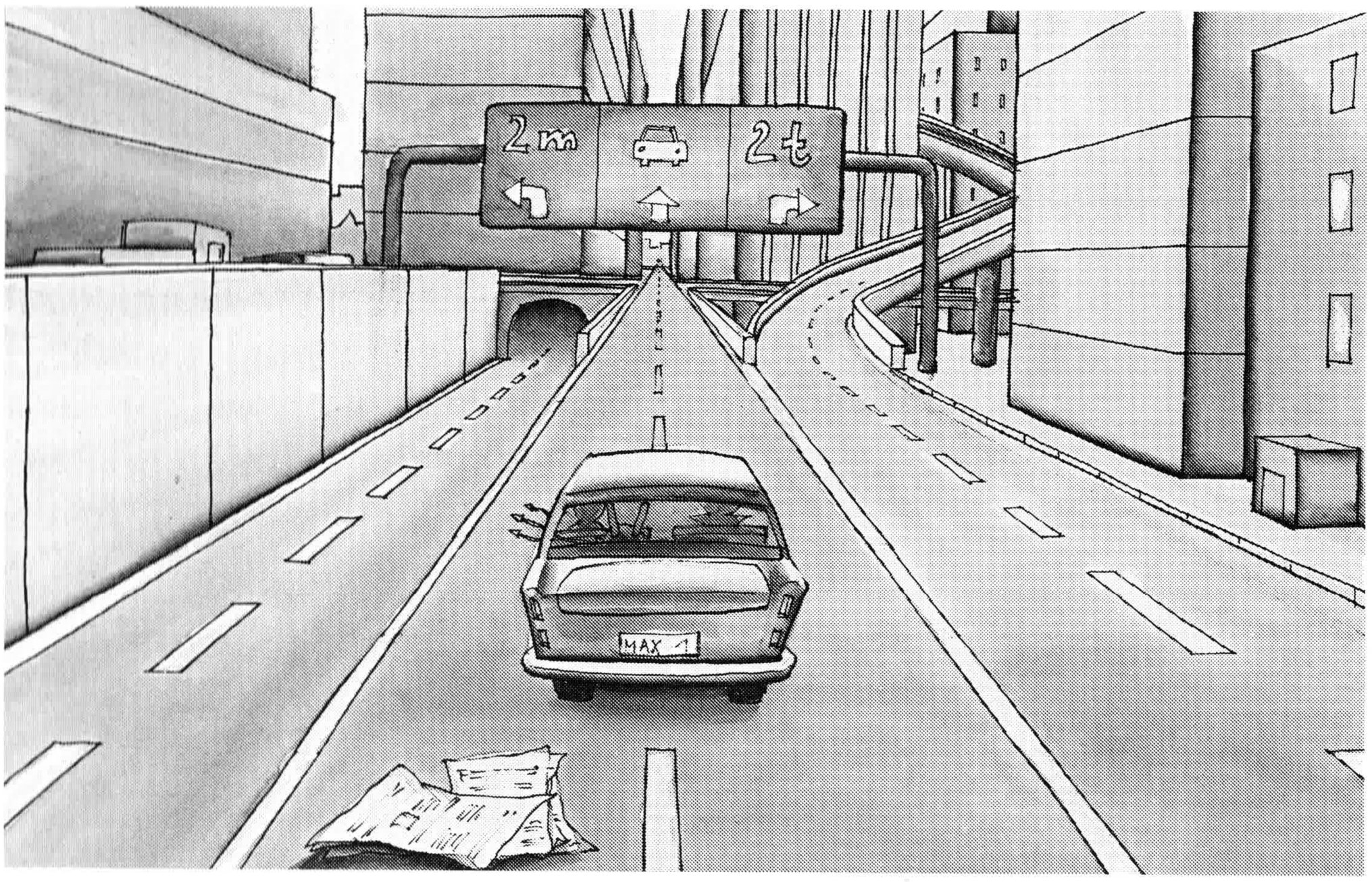
```
10 PRINT "DARF ICH FRAGEN, IN WELCHEM JAHR"
20 PRINT "SIE GEBOREN SIND?"
30 INPUT A
40 PRINT "UND WELCHES JAHR HABEN WIR JETZT?"
50 INPUT B
60 PRINT:PRINT "DANN SIND SIE JETZT ";B-A; JAHRE ALT."
```

Das Programm ist leicht zu durchschauen. Der Computer errechnet das Alter aus dem Unterschied (der "Differenz") zwischen dem Jahr, in dem wir uns befinden, und dem Jahr, in dem der Benutzer des Programms geboren ist. Diese Differenz ergibt nichts anderes als das Alter des Benutzers. Wenn du es nicht glaubst, probiere es aus und gib dein eigenes Geburtsjahr ein.

Wir wollten den Computer jedoch Komplimente machen lassen. Diese Komplimente sollten vom Input abhängen. Das heißt, sie hängen von den beiden Daten ab, aus denen der Computer das Alter des Benutzers errechnet hat. Dazu verwenden wir wieder die "bedingte Verzweigung", nämlich

IF . . .THEN

Diese Verzweigung könnten wir nach dem zweiten Input einbauen. Denn das ist die Stelle, an der der Computer schon alle Informationen erhalten hat, um seine Aufgabe lösen zu können.



Ist der Benutzer jünger als zwölf Jahre, dann soll der Computer "Du" statt "Sie" zu ihm oder zu ihr sagen:

```
55 IF B-A<=12 THEN PRINT "DANN DARF ICH
WOHL NOCH DU SAGEN":GOTO 70
```

Im Klartext: *Wenn* das Alter kleiner oder gleich 12 ist, *dann* soll der Computer dem Benutzer mitteilen, daß er ab jetzt "DU" zu ihm sagt. Der Befehl GOTO 70 ist notwendig, damit das Programm in diesem Fall die Zeile 60 überspringt; denn dort wird der Benutzer ja wieder mit "SIE" angesprochen. Zeile 70 lautet folglich

```
70 PRINT "DU BIST";B-A;" JAHRE ALT."
```

Bei dieser Zeile mußt du vor und nach der vom Computer errechneten Zahl Leerzeichen lassen, also innerhalb der Anführungszeichen. Dadurch sorgst du im Output-Text für die nötigen Zwischenräume. Du kannst ja mal ausprobieren, wie das Ergebnis aussehen würde, wenn man sie wegläßt.

Nun wollen wir aber, wie wir uns vorgenommen haben, den Computer ein paar Komplimente machen lassen, und zwar entsprechend dem Alter des Benutzers. Schauen wir uns in Abbildung HC 17 den Vorschlag für ein solches Programm an:

- In der ersten Zeile haben wir unserem Programm einen Namen gegeben. Unter diesem Namen kann man es auf Diskette oder Kassette auch abspeichern.
- Nachdem der Computer den Bildschirm gelöscht hat, begrüßt er den Benutzer und bittet ihn um die Informationen.
- Aus den beiden eingegebenen Zahlen errechnet er das Alter des Benutzers. Die "Verarbeitung" dieser Daten geschieht aber nun anders als in unserem ersten Programm. Sie ist in den Zeilen 90 bis 100 mit dem Sortieren der Ergebnisse verbunden. Für vier verschiedene Möglichkeiten bietet das Programm einen jeweils anderen Output.
- In Zeile 120 benötigen wir keine IF/THEN-Bedingung mehr, denn jetzt kann nur noch B-A größer als 40 sein.

Das Programm funktioniert allerdings nur, wenn hinter den einzelnen Outputs ein END kommt. Sonst würde der Computer

```
10 REM ***KOMPLIMENTE***
20 PRINT "␣GUTEN TAG."
30 PRINT "DARF ICH FRAGEN,"
40 PRINT "IN WELCHEM JAHR SIE GEBOREN SIND"
50 INPUT A
60 PRINT:PRINT"UND WELCHES JAHR"
70 PRINT "HABEN WIR JETZT"
80 INPUT B
90 IF B-A<=16 THEN PRINT "␣DU SIEHST SEHR ERWACHSEN AUS FUER DEIN  ALTER.":END
100 IF B-A<=28 THEN PRINT "␣EIN FLOTTER TWEN WIE SIE KOENNTE MIR GE-FALLEN.":END
110 IF B-A<=40 THEN PRINT "␣MAN SIEHT GLEICH:SIE SIND IN DEN BESTEN JAHREN.":END
120 PRINT "␣SIE SEHEN WIRKLICH GUT AUS FUER IHR     ALTER.":END
```

Abb. HC 17

auch immer die nächste Programmzeile abarbeiten. Jemand, der 14 Jahre alt ist, würde dann alle vier Komplimente nacheinander bekommen.

Allerdings ließe sich statt dem Befehl END ebensogut ein GOTO verwenden. Zum Beispiel könnten wir das Programm nach dieser Verzweigung in eine Zeile mit der Nummer 130 schicken; in der wird dann beispielsweise ein anderes Programm geladen oder ein Rechenspiel begonnen.

Der Commodore 64 reagiert allerdings etwas heikel auf die Textmenge in einer Programmzeile. Manchmal bleibt einem nichts anderes übrig, als den Text zu kürzen. Oder man fügt eine weitere Textzeile ein. Jedoch muß man dann auch dafür sorgen, daß die zusätzliche Textzeile ausschließlich von der bedingten Verzweigung angesprochen wird.

Das Programm in Abbildung HC 18 enthält einen Fehler; kannst du ihn beheben?

```
10 PRINT "RATE EINE ZAHL"
20 PRINT "ZWISCHEN 1 UND 10"
30 INPUT A
40 IF A<>3 THEN PRINT "TUT MIR LEID."
60 PRINT "NOCH EIN VERSUCH!":GOTO 30
70 PRINT "HERZLICHEN GLUECKWUNSCH!"
80 PRINT "DIE 3 WAR GESUCHT.":END
```

Abb. HC 18

Die Lösung (Abbildungen HC 19—HC 21) haben wir auf den Kopf gestellt.

Übrigens: In unserer Fassung des "Komplimente"-Programms aus Abbildung HC 17 wurden im Text an manchen Stellen Lücken gelassen. Und an einer Stelle steht mitten im Wort ein Trennungsstrich; warum wohl?

Richtig, damit verhindern wir, daß der Text über die Bildschirmzeile hinaus geschrieben wird und der Computer die Wörter ohne Trennungszeichen trennt. Das würde zwar den Programmablauf nicht weiter stören; und lesen könnte man den Text immer noch. Aber es sieht doch schöner aus.

Wo ein Trennungsstrich eingefügt werden muß, läßt sich einfach ausrechnen. Du weißt ja: In eine Bildschirmzeile passen insgesamt vierzig Buchstaben oder Leerzeichen. Für größere Textmengen ist es ratsam, du zeichnest dir wieder eine "Käst-

```
10 PRINT "RATE EINE ZAHL"
20 PRINT "ZWISCHEN 1 UND 10"
30 INPUT A
35 IF A=3 THEN GOTO 70
40 PRINT "TUT MIR LEID."
60 PRINT "NOCH EIN VERSUCH!":GOTO 30
70 PRINT "HERZLICHEM GLUECKWUNSCH!"
80 PRINT "DIE 3 WAR GESUCHT.":END
READY.
```

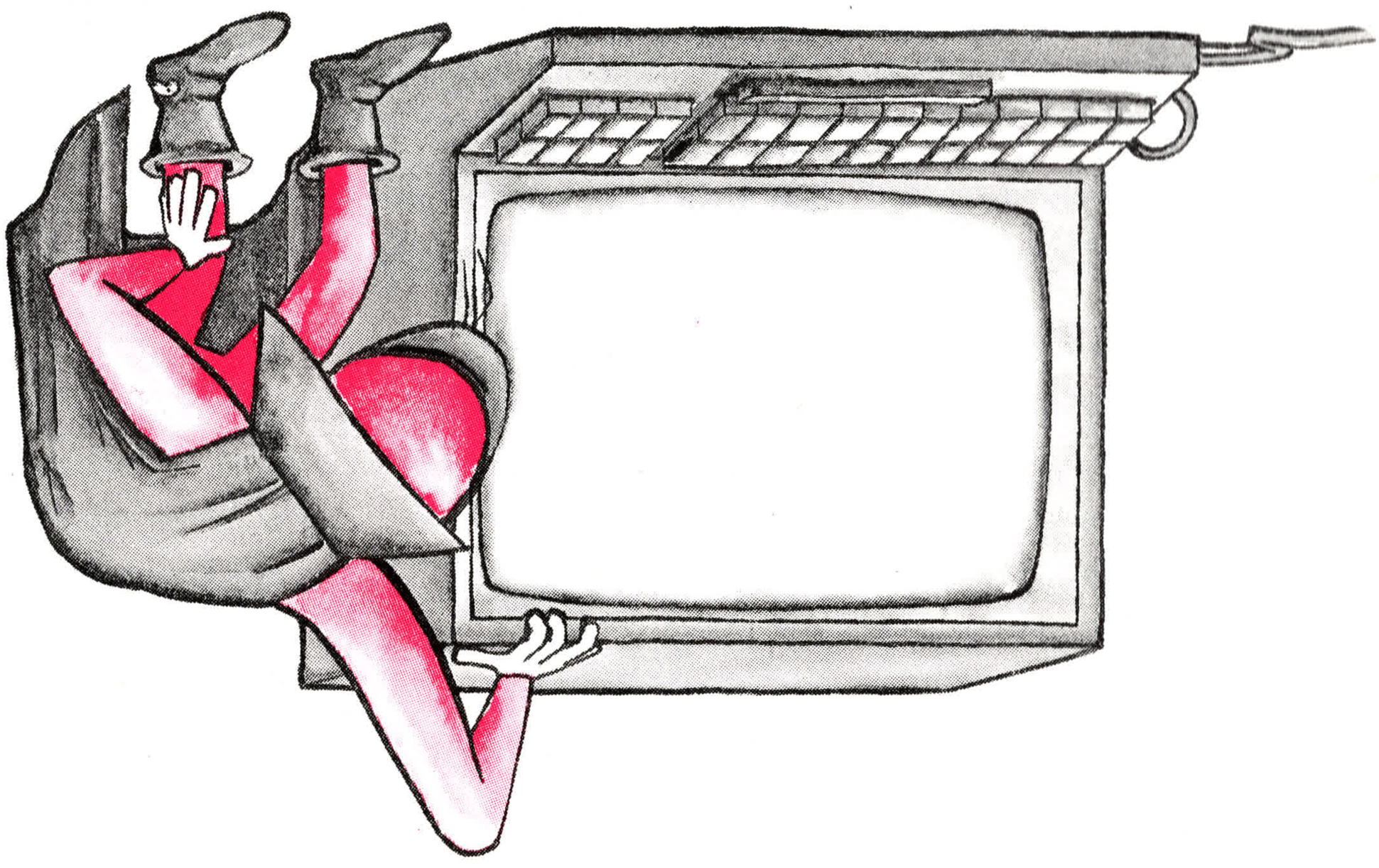
Abb. HC 19

```
10 PRINT "RATE EINE ZAHL"
20 PRINT "ZWISCHEN 1 UND 10"
30 INPUT A
40 IF A<>3 THEN PRINT "TUT MIR LEID."
60 IF A<>3 THEN PRINT "NOCH EIN VERSUCH!":GOTO 30
70 PRINT "HERZLICHEN GLUECKWUNSCH!"
80 PRINT "DIE 3 WAR GESUCHT.":END

READY.
```

Abb. HC 20

```
10 PRINT "RATE EINE ZAHL"
20 PRINT "ZWISCHEN 1 UND 10"
30 INPUT A
40 IF A<>3 THEN PRINT "TUT MIR LEID.":GOTO 90
70 PRINT "HERZLICHEN GLUECKWUNSCH!"
80 PRINT "DIE 3 WAR GESUCHT.":END
90 PRINT "NOCH EIN VERSUCH!":GOTO 30

READY.
```

Abb. HC 21



chen-Version“ vom Bildschirm. Das kann so aussehen wie in Abbildung Z 1.

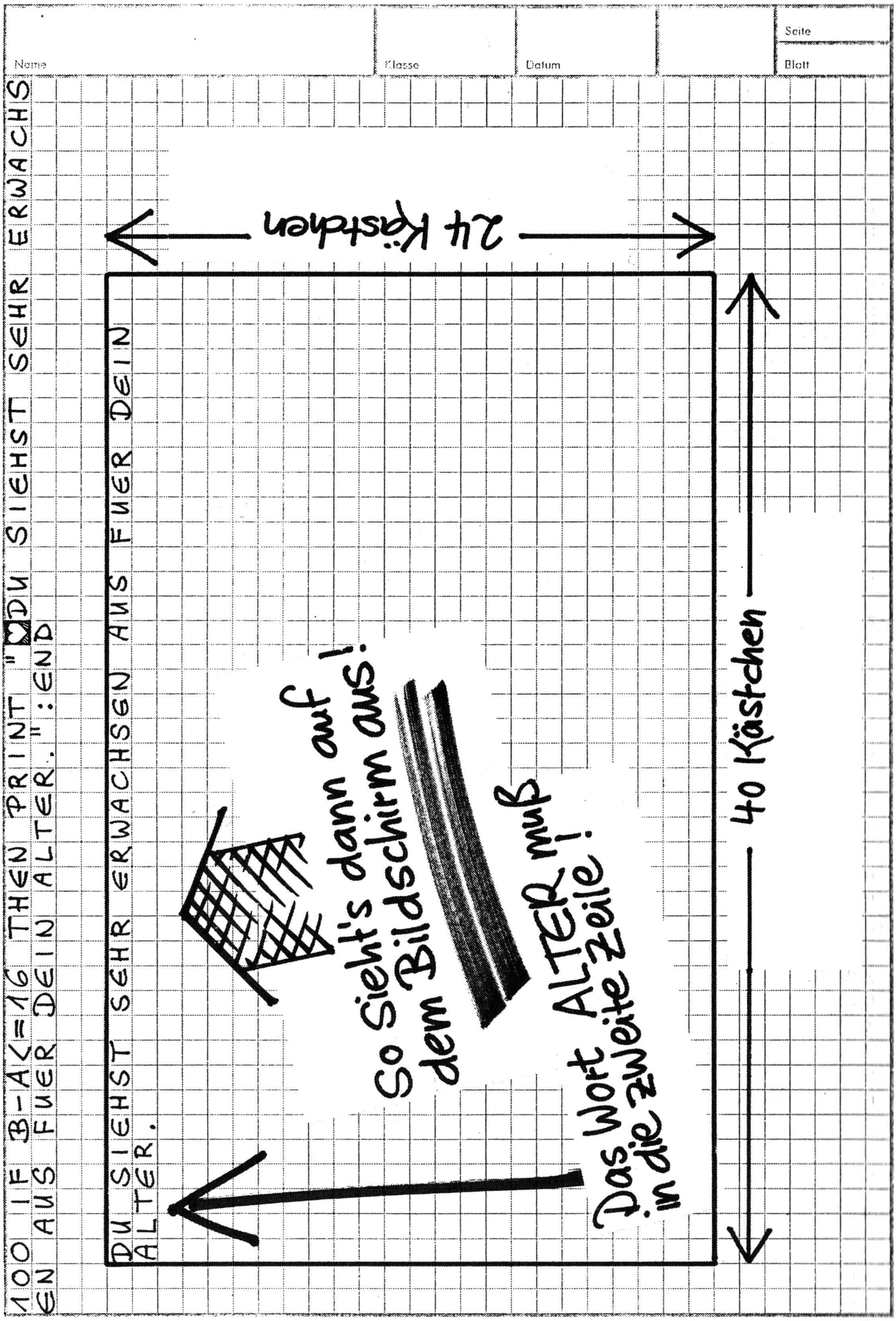


Abb. Z 1

Durch Ausprobieren und Ändern direkt auf dem Bildschirm kommt man ebensogut auf die richtigen Stellen zum Zeilentrennen.

In allen unseren "EVA"-Programmen sind wir nach dem gleichen Muster vorgegangen:

a) Eine PRINT-Zeile oder mehrere PRINT-Zeilen sagen dem Benutzer, was ihn erwartet und welche Eingabe von ihm verlangt wird.
b) Eine INPUT-Zeile oder mehrere INPUT-Zeilen nehmen die Eingabe auf.
c) Der Computer verarbeitet und sortiert die eingegebenen Daten, zum Beispiel durch eine bedingte Verzweigung mit Hilfe von IF . . .THEN oder mehrerer solcher Verzweigungen.
d) Der Computer gibt in einer PRINT-Zeile das Ergebnis aus.

Wir haben schon etliche Programme geschrieben, in denen die eingegebenen Daten in ein und derselben Zeile verarbeitet und ausgegeben werden. Etwa dort, wo in einer Zeile auf das IF . . .THEN ein PRINT folgt. Bei kürzeren Informationen für den Benutzer kann man auch Kommentar und INPUT in einer Zeile unterbringen. Dazu schreibst du den Text in Anführungszeichen unmittelbar hinter den INPUT-Befehl.

Statt in zwei Programmzeilen wie

```
10 PRINT "BITTE EINE ZAHL"
20 INPUT A
```

kann man platzsparender schreiben:

```
10 INPUT "BITTE EINE ZAHL";A
```

Indem nach dem Text ein Semikolon (;) gesetzt wird, erreicht man, daß Text und Eingabe auch auf dem Bildschirm in einer Zeile stehen. Mit einem anderen Zeichen als dem Semikolon funktioniert es allerdings nicht! Dieser Trick kommt uns vor allem dann zugute, wenn uns an übersichtlichen Eingaben gelegen ist.

Als Beispiel soll ein kleines Programm zum Addieren ("Zusammenzählen") dienen, wie es in Abbildung HC 22 zu sehen ist.

Was nach dem Programmstart auf dem Bildschirm geschieht, siehst du in Abbildung HC 23.

```
10 PRINT "ICH KANN ZWEI ZAHLEN ADDIEREN"
20 INPUT "ERSTE ZAHL";A
30 INPUT "ZWEITE ZAHL";B
40 PRINT A;" + ";B;" = ";A+B
```

Abb. HC 22

```
ICH KANN ZWEI ZAHLEN ADDIEREN
ERSTE ZAHL? 123
ZWEITE ZAHL? 45
 123  +  45  =  168

READY.
```

Abb. HC 23

Schon bei unserem "Einmaleins-Programm" konnten wir zum EVA-Schema eine weitere Stufe hinzufügen. Wir haben es nämlich gegen unerwünschte Eingaben "geschützt". Das war in dem Fall zwar eher ein Programmier-Gag, um dem Benutzer vorzutäuschen, daß der Computer "denkt". Doch oft ist solch ein Programmschutz gegen unerwünschte Eingaben unerläßlich.

Auch hierbei geht man meist nach dem gleichen Schema vor:

Zuerst wird dem Benutzer gesagt, was er eingeben soll. Das erreichen wir mit der folgenden Programmzeile:

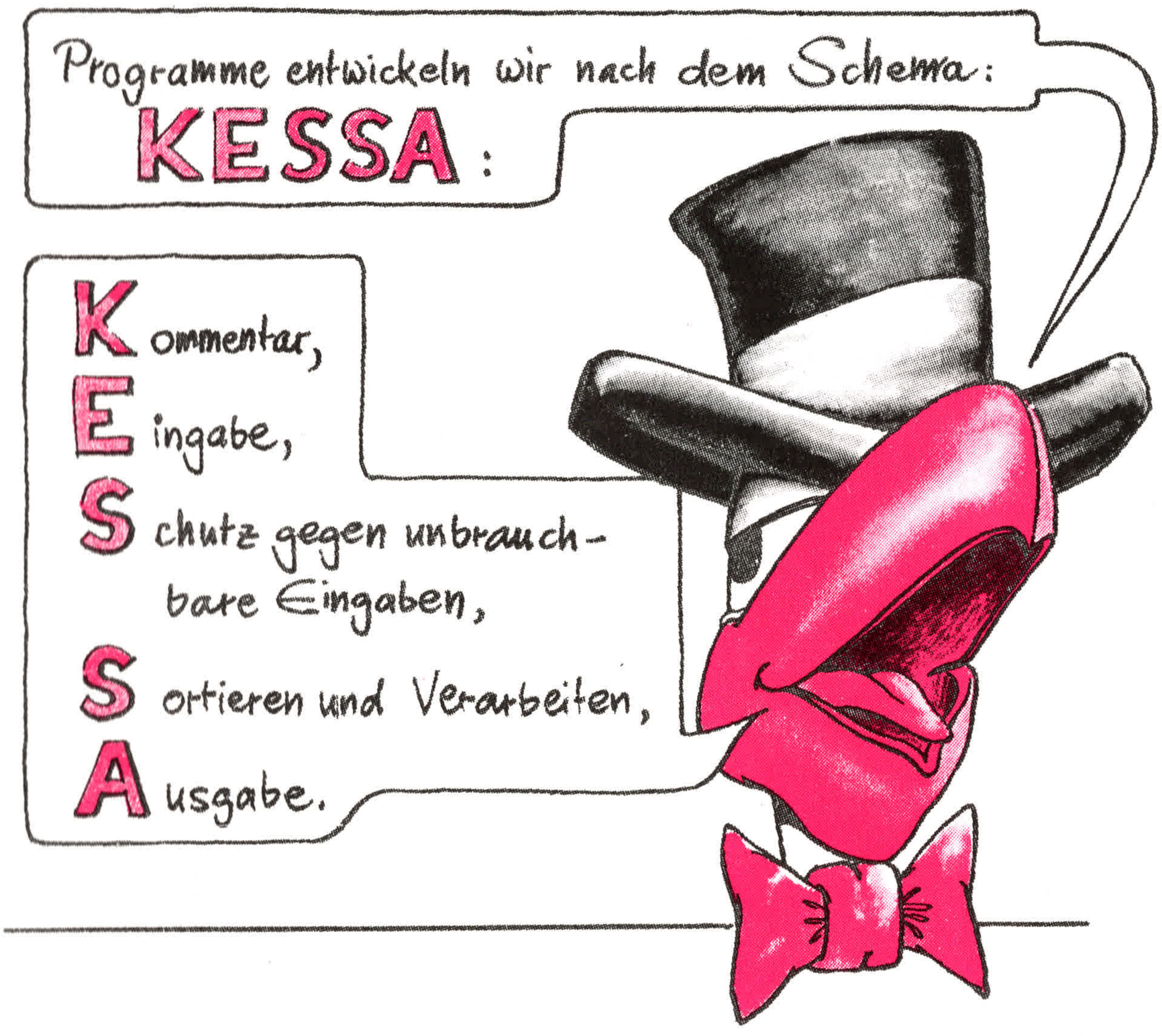
```
10 INPUT "BITTE EINE ZAHL UNTER 10";A
```

Bevor nun das Programm die eingegebenen Daten verarbeitet, soll es prüfen, ob die Eingabe auch jene Bedingung erfüllt, die in Zeile 10 aufgestellt worden ist. Und wenn die Bedingung nicht erfüllt ist, dann soll das Programm eine neue Eingabe verlangen.

Die Programmzeile dazu kann so aussehen:

```
20 IF A>=10 THEN PRINT "FALSCHE
EINGABE!":GOTO 10
```

Nun verfügen wir über ein vollständiges Schema für eine Art von Programmen, aus denen eine Menge herauszuholen ist. Nennen wir es einmal das KESSA-Prinzip. Zwar wirst du dir unter dem Wort allein zunächst wohl kaum etwas vorstellen können. Aber man kann es sich zweifellos gut merken.



Mit diesem Modell und anhand der BASIC-Befehle, die wir bis jetzt ausprobiert haben, kannst du nun bestimmt selbständig einige interessante Programme entwerfen. Vielleicht versuchst du dich einmal an einem Programm, das die Zahlen 1,2,3,4,5 in die Worte "eins","zwei","drei","vier" und "fünf" übersetzt. Wie ein derartiges Programm aussehen könnte, siehst du in Abbildung HC 24.

```
10 REM *ZAHLEN IN WORTE*
20 PRINT "BITTE GIB EINE ZAHL"
30 PRINT "ZWISCHEN  >1< UND >5<"
40 INPUT "EIN";A
50 IF A>5 THEN PRINT"KANNST DU NICHT LESEN?":GOTO 20
60 PRINT "DAS WAR EINE";
70 IF A=1 THEN PRINT " EINS":END
80 IF A=2 THEN PRINT " ZWEI":END
90 IF A=3 THEN PRINT " DREI":END
100 IF A=4 THEN PRINT " VIER":END
110 IF A=5 THEN PRINT " FUENF":END
```

Abb. HC 24

Oder du entwirfst ein Programm, das nach Eingabe der entsprechenden Zahlen den Namen des Wochentages ausgibt. Gefragt wird etwa "DER WIEVIELTE TAG DER WOCHE IST

HEUTE?“. Nachdem du die Zahl 3 eingegeben hast, könnte der Output lauten:

HEUTE IST MITTWOCH.

In unseren Lösungsvorschlägen am Ende dieses Kapitels haben wir die Wochentage in mehreren Sprachen ausgeben lassen. Vielleicht gelingt es dir auch schon zumindest mit den englischen Wochentagen.

Oder wie wäre es mit folgendem Programm: Der Benutzer soll eine Zahl zwischen 1 und 20 raten. Der Computer sagt dem Ratekandidaten, ob es ”heiß“, ”warm“, ”lau“, ”kühl“, ”kalt“ oder ”eiskalt“ ist, also ob die Eingabe nahe an der richtigen Lösung dran ist oder weit davon entfernt.

Lösungsvorschläge für diese Programm-Ideen findest du am Ende dieses Kapitels. Versuche zunächst, die Lösung auf eigene Faust zu finden. Denke dran: für ein Programmierproblem kann es stets unterschiedliche Lösungen geben. Daher ist es durchaus üblich, daß die Programme ganz verschieden aussehen können, auch wenn sie jeweils dasselbe zu bieten haben.

Jedenfalls fänden wir es toll, wenn du nach unserem KESSA-Prinzip eigene Programme schreibst.

Falls du Lust hast, schicke uns doch deine Programm-Ideen zu. Auf Kassette, Diskette oder als ausgedrucktes beziehungsweise handgeschriebenes Listing. Notfalls genügt auch eine Programmbeschreibung.

Besonders interessieren uns Programme, die in diesem und den vorhergehenden Büchern noch nicht besprochen wurden.

Die besten Programm-Ideen belohnen wir mit Bücherpreisen. Vielleicht drucken wir auch das eine oder andere einmal ab.

Hier ist unsere Adresse:

Computer kinderleicht
Franz Schneider Verlag
Frankfurter Ring 150
8000 München 46

Wir warten auf die Super-KESSA- und sonstigen Programme!!!

Merkst du etwas? In Zeile 90 mußte ich mit den Trennungszeichen etwas "schwindeln"

```
10 INPUT "␣DER WIEVIELTE TAG DER WOCHE IST HEUTE";A
20 PRINT "HEUTE IST"
30 PRINT" ┌────┐","┌────┐","┌────┐","┌────┐"
40 PRINT" |  D |","| GB|","|  F |","|  I |"
50 PRINT" └────┘","└────┘","└────┘","└────┘"
60 IF A=1 THEN PRINT"MONTAG","MONDAY","LUNDI","LUNEDI"
70 IF A=2 THEN PRINT "DIENSTAG","TUESDAY","MARDI","MARTEDI"
80 IF A=3 THEN PRINT "MITTWOCH","WEDNESDAY","MERCREDI","MERCOLEDI"
90 IF A=4 THEN PRINT "DONNERSTAG ";"THURSDAY","JEUDI","GIOVEDI"
100 IF A=5 THEN PRINT "FREITAG","FRIDAY","VENDREDI","VENERDI"
110 IF A=6 THEN PRINT "SAMSTAG","SATURDAY","SAMEDI","SABATO"
120 IF A=7 THEN PRINT "SONNTAG","SUNDAY","DIMANCHE","DOMENICA"
```

Abb. HC 25

Und so läuft das Programm auf dem Bildschirm:

```
DER WIEVIELTE TAG DER WOCHE IST HEUTE? 4
HEUTE IST
 ┌────┐    ┌────┐    ┌────┐    ┌────┐
 |  D |    | GB|    |  F |    |  I |
 └────┘    └────┘    └────┘    └────┘
DONNERSTAG THURSDAY JEUDI     GIOVEDI

READY.
```

Abb. HC 26

```
10 PRINT "■■ES GEHT DARUM, EINE ZAHL"
20 PRINT "ZWISCHEN >1< UND >20<"
30 PRINT "ZU ERRATEN."
40 PRINT "■■■■ICH SAGE DIR,"
50 PRINT "OB DU NAHE DRAN BIST"
60 PRINT "ODER WEIT VON DER LOESUNG ENTFERNT."
70 PRINT "■■■■WENN DU FERTIG BIST,"
80 PRINT "DRUECKE DIE TASTE MIT DER >1<."
90 INPUT A:IF A=1 THEN GOTO 120
100 PRINT "■WAS IST LOS? IST DOCH GANZ EINFACH!"
110 PRINT "■■ES GEHT DARUM, EINE ZAHL":GOTO 20
120 INPUT "DEINE ZAHL";B
130 IF B>20 THEN PRINT "GILT NICHT!":GOTO 100
140 IF B<1 THEN PRINT "GILT NICHT!":GOTO 100
150 IF B=11 THEN PRINT "■■■■/\/\/\/\RICHTIG/\/\/\/":END
160 IF B>11 THEN GOTO 210
170 IF 11-B<3 THEN PRINT "■HEISS!":GOTO 120
180 IF IF 11-B<5 THEN PRINT "■WARM!":GOTO 120
190 IF 11-B<7 THEN PRINT "■LAU!":GOTO 120
200 IF 11-B<=10 THEN PRINT "■KALT!":GOTO 120
210 IF B-11<3 THEN PRINT "■HEISS!":GOTO 120
220 IF B-11<5 THEN PRINT "■WARM!":GOTO 120
230 IF B-11<7 THEN PRINT "■LAU!":GOTO 120
240 IF B-11<=9 THEN PRINT "■KALT!":GOTO 120

READY.
```

Abb. HC 27

Ein Ladeprogramm mit Überraschungen und der C 64 als Quasselkasten

Vielleicht hast du schon öfter Disketten- oder Kassettenprogramme geladen. Möglicherweise hast du auch schon mit "Ladeprogrammen" zu tun gehabt. Jedenfalls sind dies Programme, die sich mit einem

READY

melden, nachdem sie geladen worden sind. Sobald man sie dann mit

RUN

startet, wird erst das eigentliche Hauptprogramm geladen. Diese Ladeprogramme sind meist in BASIC geschrieben und die Hauptprogramme in Maschinensprache.

Grundsätzlich läßt sich jedem Programm ein extra Ladeprogramm vorausstellen. Wie wir ja wissen, lautet die Anweisung zum Laden eines Programms von der Kassette

LOAD "....."

und zum Laden von der Diskette

LOAD ".....",8

Diese Ladeanweisung kannst du nicht nur im Direkt-Modus (also ohne Programmzeilen) eingeben, sondern auch innerhalb eines normalen BASIC-Programms. Somit tippen wir beispielsweise zum Laden von der Diskette nicht das gewohnte

LOAD "MAX'EINMALEINS",8

sondern du gibst statt dessen folgende Programmzeile ein:

```
10 LOAD "MAX'EINMALEINS",8
RUN
```

Warum nicht gleich mit einem der Programme auf deinen Disketten (oder Kassetten) ausprobieren?

In einem derartigen Ladeprogramm könnte man auch wieder

Fragen an den Benutzer unterbringen. Schreiben wir dazu entsprechend unserem Modell ein kleines Programm.

Bevor wir einfach drauflostippen, einige besonders wichtige Hinweise: Wer ein Programm schreiben will, muß sich zuvor genau überlegen, was das Programm bewirken soll. Danach legt man fest, wie es in BASIC übersetzt wird, und in welchen Schritten man dabei vorgeht.

Unser nächstes Programm soll den Benutzer zuerst fragen, ob er das Einmaleins-Programm von Max tatsächlich benutzen will oder nicht. (Es befindet sich übrigens auch auf der Diskette zu diesem Buch.)

Je nachdem, was der Benutzer als INPUT eingibt, soll der Computer entweder das Programm laden oder sich verabschieden.

Das bedeutet: Als erstes erscheint die Bemerkung. Sie informiert den Benutzer darüber, was ihm geboten wird und stellt ihm die entscheidende Frage.

Das Programm könnte zum Beispiel fragen:

”WILLST DU MIT DEM EINMALEINS-PROGRAMM
VON MAX UEBEN?“

Der Benutzer soll Gelegenheit erhalten, dem Computer seine Entscheidung mitzuteilen. Man könnte ihn zum Beispiel zwischen ”JA“ und ”NEIN“ entscheiden lassen. Am einfachsten ist es, wenn er für ”Ja“ eine ”1“ eintippen kann und eine ”0“ für ”NEIN“.

Doch auch das muß der Computer dem Benutzer vorher mitteilen. Dies bedeutet noch etwas mehr Text. Daher müssen wir ein paar PRINT-Zeilen vor den eigentlichen INPUT setzen. Diesem Input könnten wir den Variablen-Namen E (für ”Entscheidung“) geben. Natürlich ginge auch jeder andere Buchstabe, auch jede Kombination aus zwei Buchstaben oder gar Kombinationen aus einem Buchstaben und einer Zahl.

Nachdem der Benutzer seinen INPUT eingetippt hat, müssen wir das Programm wieder gegen unerwünschte Eingaben schützen. In Worten ausgedrückt hieße die dazugehörige BASIC-Zeile:

Wenn der eingegebene Wert kleiner als Null oder größer als Eins ist, **dann** soll der Computer sagen, daß er diese Eingabe nicht versteht. Daraufhin soll der Benutzer die Eingabe wiederholen. (Zahlen kleiner als Null sind jene mit dem Minuszeichen

davor, also die negativen Zahlen, von denen wir schon im letzten Band gesprochen haben.)

Diese Bedingung in BASIC zu übersetzen, fällt dir wahrscheinlich auch nicht schwerer als beim ersten Teil des Programms. Selbstverständlich gibt es von uns wieder am Ende dieses Kapitels einen "Spickzettel", also einen Lösungsvorschlag.

Nun kommt der eigentliche Sortier-Vorgang. Das Programm unterscheidet zwischen den beiden einzigen verbliebenen Möglichkeiten des INPUT: **Wenn** die eingegebene Zahl gleich Null ist, **dann** soll der Computer sich höflich verabschieden und das Programm beenden. **Wenn** die eingegebene Zahl gleich Eins ist, **dann** soll der Computer das Programm "MAX'EINMAL-EINS" laden.

Falls du dieses Programm gemäß unserer "Programmbeschreibung" geschrieben hast, und es läuft nach dem Starten nicht gleich so, wie du es dir vorstellst, gib bitte nicht gleich auf. Suche erst nach dem Fehler. Ohne Fehlersuche bringt zuweilen auch der routinierteste Programmierer sein Programm nicht in Gang. Hat sich ein "Wurm" ins Programm geschlichen, nennt der Profi einen solchen Fehler "Bug" (sprich "Bag"). Und die systematische Jagd auf den Bug nennt man im Englischen "Debugging" (sprich "dibagging"). Die Meldung

?SYNTAX ERROR IN _____ (hier steht eine Zeilennummer)

bedeutet: Jetzt geht's nicht ohne "Debugging". Damit sagt uns der Computer, daß ein fehlerhafter BASIC-Befehl in der angegebenen Programmzeile steht. Das kann manchmal nur ein "Tippfehler" sein – etwa wenn man statt PRINT ein PRIMT eingetippt hat. Das passiert immer wieder und ist nicht schwer herauszufinden.

Schwieriger wird es schon, falls wir falsche Satzzeichen bei den BASIC-Begriffen verwendet haben. Denn die fallen nicht so leicht auf. Zum Beispiel versteht der Computer die folgende Programmzeile ohne weiteres richtig (schließlich stimmt sie ja auch!):

```
70 IF E=1 THEN PRINT "SCHOEN,":LOAD
"MAX'EINMALEINS",8
```

Aber in der nun folgenden Programmzeile hat sich ein "bug"

eingenistet. Diese Zeile versteht der Computer nämlich nicht:

```
70 IF E=1 THEN PRINT "SCHOEN!":LOAD
"MAX'EINMALEINS".8
```

Auf geht's zum Debugging! Die kleinsten Ursachen haben unübersehbare Auswirkungen. Der Computer schimpft zwar nicht. Aber er läßt nicht einen einzigen Fehler durchgehen, und sei es ein noch so winziger Kommafehler.

Außer den BASIC-Fehlern gibt es auch "Programm-Fehler". Dann arbeitet der Computer das Programm zwar ohne Fehlermeldungen ab, aber er kommt dabei nicht zum gewünschten Ergebnis.

Nehmen wir an, du vergißt an einer wichtigen Stelle unseres Programmes den Befehl

```
END
```

Der Computer reagiert dann ziemlich unsinnig. Sobald man nämlich die Null für "NEIN" eingibt, erscheint als Ergebnis ungefähr so etwas:

```
"NUN, VIELLEICHT EIN ANDERES MAL. AUF
WIEDERSEHEN!"
"SCHOEN! ICH LADE GLEICH DAS PROGRAMM."
```

Und jetzt wird das Programm geladen, ohne daß es der Benutzer wollte!

Ich wette, daß dein Programm nach ein bißchen Debugging richtig läuft. Dabei macht es nichts aus, daß es möglicherweise etwas anders aufgebaut ist als unser Lösungsvorschlag. Die Abbildungen HC 28 und HC 29 zeigen unsere Lösungsvorschläge.

Nichts spricht dagegen, daß du dieses Programm noch weiter ausgestaltest. Du willst vielleicht mehr Kommentare einbauen, die Bedingungen verändern und so weiter.

Wir entwickeln jetzt ein neues Ladeprogramm. Es bietet mehr als nur zwei Eingabemöglichkeiten. Überdies spielt diesmal für den lieben Benutzer auch der Zufall eine Rolle. Daß er das nächste Programm laden darf, muß er sich nämlich erst einmal verdienen!

Bevor er das Programm laden kann, soll der "Anwender" aus zehn Zahlen eine auswählen. Nur eine von den zehn Zahlen ist

```
10 PRINT "GUTEN TAG,"
20 PRINT "WILLST DU MIT DEM EINMALEINS-PROGRAMM"
30 PRINT "VON MAX UEBEN?"
40 PRINT"FUER >JA< TIPPE DIE >1<."
50 PRINT "FUER >NEIN< TIPPE DIE >0<."
60 INPUT E:IF E<0 OR E>1 THEN PRINT "VERSTEHE ICH NICHT!":GOTO 40

READY.
```

Abb. HC 28

```
10 PRINT "GUTEN TAG,"
20 PRINT "WILLST DU MIT DEM EINMALEINS-PROGRAMM"
30 PRINT "VON MAX UEBEN?"
40 PRINT"FUER >JA< TIPPE DIE >1<."
50 PRINT "FUER >NEIN< TIPPE DIE >0<."
60 INPUT E:IF E<0 OR E>1 THEN PRINT "VERSTEHE ICH NICHT!":GOTO 40
70 PRINT "WIE SCHOEN!ICH LADE GLEICH DAS PROGRAMM.":LOAD "MAX' EINMALEINS",8

READY.
```

Abb. HC 29

die richtige. Erst wenn er die herausgefunden und eingetippt hat, soll der Computer das gewünschte Programm laden. Andernfalls bringt er irgendeine "Ausrede" oder einen Allerweltssatz auf den Bildschirm und fragt noch einmal nach der richtigen Zahl.

Wir beginnen wieder mit dem Kommentar und dem INPUT. Anschließend schützen wir das Programm gegen unliebsame Eingaben. Und daraufhin kann man sich für neun von den zehn möglichen Eingabezahlen jeweils irgendeinen mehr oder weniger schlauen Satz einfallen lassen. Bei einer dieser Zahlen soll das Programm ihm allerdings bestätigen, daß er richtig geraten hat und das Programm schließlich laden.

Unseren Vorschlag für dieses Programm siehst du in Abbildung HC 30.

Auch hier gilt: Dein Programm kann sehr wohl ganz anders aussehen und trotzdem funktionieren.

In Zeile 10 haben wir mit Hilfe des Befehls PRINT einige Leerzeilen eingefügt, damit es auf dem Bildschirm etwas übersichtlicher zugeht. Nach jedem "Durchlauf", bei dem die richtige Zahl nicht getroffen wurde, kommt das Programm ja zu dieser Zeile zurück.

In den Zeilen mit den Bildschirmmeldungen zu den falschen Eingaben lassen wir mit dem "inversen Herzchen" (also mit einem Steuerzeichen!) stets den Bildschirm löschen. Dadurch kommen die "Kommentarzeilen" besser zur Geltung. Jeder einzelne Satz wirkt so, als hätte ihn der Computer sich gerade "ausgedacht". Der Befehl GOTO 140 in Zeile 120 soll zu jener Zeile führen, in der das LOAD untergebracht ist. Ebensogut könnte man dort auch das Programm selbst anfügen, oder aber neue "Abfragen" formulieren, zum Beispiel welche "Version" eines Spieles man spielen will. Oder wir könnten für die Zeit des Ladevorgangs eine Bildschirmgrafik aufbauen.

Bevor wir unser nächstes Programm entwickeln, möchte ich eine Geschichte erzählen:

Sie stammt aus der Zeit, als Computer noch keineswegs so selbstverständlich waren wie heute. Sie standen nur in den Laboratorien und Forschungseinrichtungen der großen Universitäten. Auch den Studenten standen nur ganz begrenzte "Rechner-Zeiten" zur Verfügung. Damals, in den sechziger Jahren, entwickelte ein amerikanischer Professor ein besonders trickreiches Programm. Es befähigte den Computer, einem

```
10 PRINT:PRINT:PRINT:PRINT"GIB IRGENDEINE ZAHL ZWISCHEN 1 UND 10   EIN."
20 INPUT A
30 IF A<1 OR A>10 THEN PRINT"ZWISCHEN 1 UND 10 HABE ICH GESAGT!":GOTO 20
40 IF A=1 THEN PRINT"␣SCHOENES WETTER HEUTE, FINDEST DU NICHT?":GOTO 10
50 IF A=2 THEN PRINT"␣AUCH MAL WIEDER AN MEINER TASTATUR. WIE NETT!":GOTO 10
60 IF A=3 THEN PRINT"␣VORSICHTIG MIT MEINEN EMPFINDLICHEN     TASTEN!":GOTO 10
70 IF A=4 THEN PRINT"␣KENNEN WIR UNS NICHT?":GOTO 10
80 IF A=5 THEN PRINT"␣SAG MAL, SIND DIE HAUSAUFGABEN SCHON    FERTIG?":GOTO 10
90 IF A=6 THEN PRINT"␣GROSSE LUST HABE ICH HEUTE EIGENTLICH   NICHT!":GOTO 10
100 IF A=7 THEN PRINT"␣WAS GIBT ES DENN HEUTE IM FERNSEHEN?":GOTO 10
110 IF A=8 THEN PRINT"␣NA JA, JETZT BIN ICH WIEDER DRAN.":GOTO 10
120 IF A=9 THEN PRINT"␣DAS WAR DIE ZAHL,GRATULIERE! ALSO WEITER":GOTO 140 *
130 IF A=10 THEN PRINT"␣IN MEINEM ROM RUMORT'S HEUTE WIEDER...":GOTO 10

READY.
```

Abb. HC 30

* Ab Programmzeile 140 beginnt das eigentliche Programm

Menschen, der ihn etwas fragte, durchaus sinnvoll zu "antworten". Jemand tippte eine Frage oder eine Aussage ein, zum Beispiel: "Mir geht es heute nicht besonders gut." Und der Computer antwortete: "Das ist aber wirklich schade, daß es Ihnen heute nicht besonders gutgeht."

Der Computer konnte aber auch zurückfragen. Zum Beispiel: "Warum geht es Ihnen heute nicht besonders gut?"

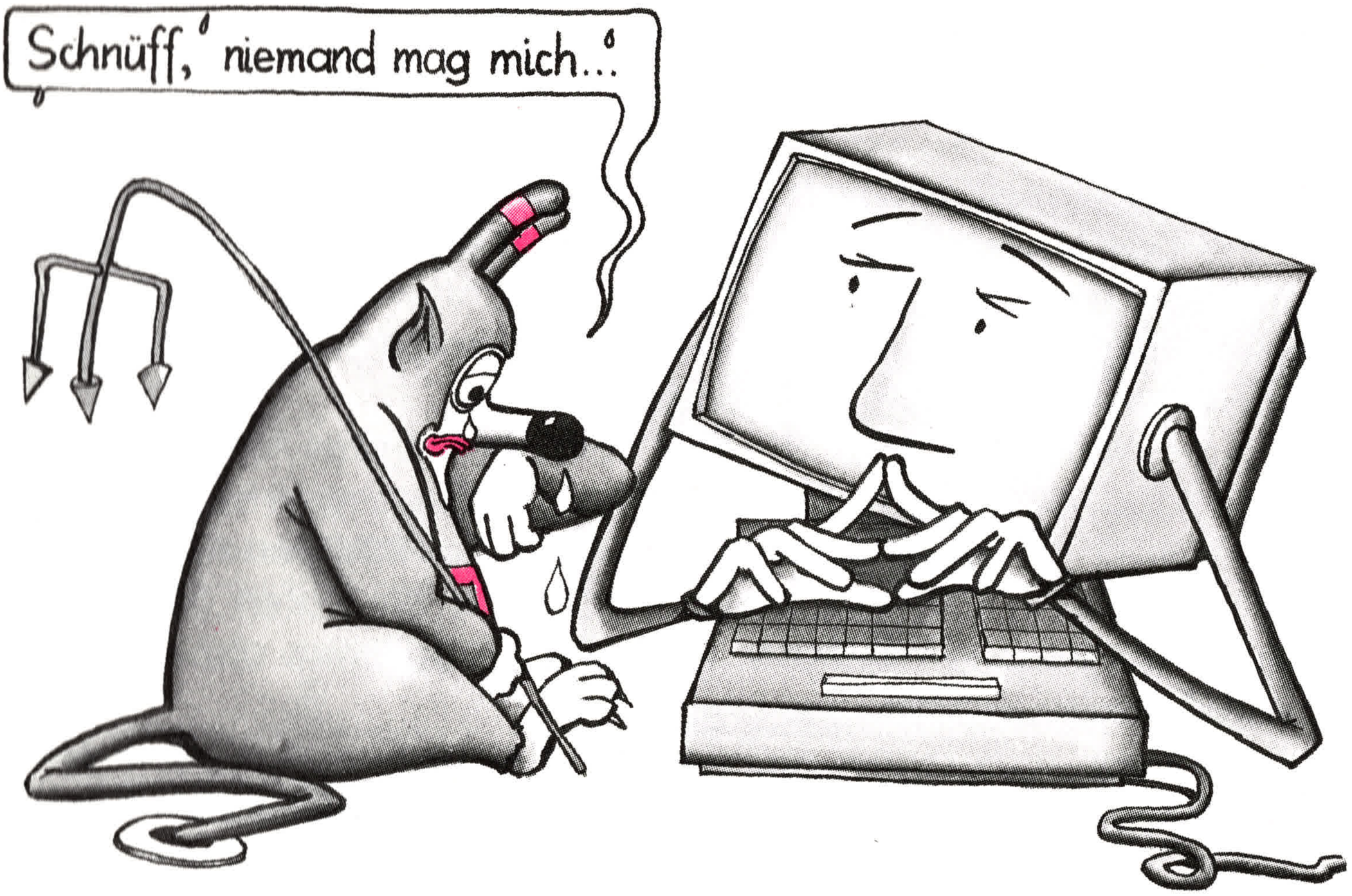
Wenn man dann antwortete: "Ich habe Kopfschmerzen", konnte der Computer wieder eine neue Antwort geben, wie: "Es tut mir sehr leid, daß Sie Kopfschmerzen haben", oder: "Warum haben Sie Kopfschmerzen?"

Das ging immer so weiter. Jeder der das Programm benutzte, hatte das Gefühl, einem besonders "verständnisvollen" Gesprächspartner gegenüberzusitzen. Das Programm war auch "schlau" genug, sich die Antworten und Eingaben zu merken. Es zählte zum Beispiel, wie oft das Wort "nein" oder "nicht" vorkam, und der Computer sagte dann zur Überraschung der Benutzer aus heiterem Himmel: "Warum sagen Sie eigentlich so oft 'nein'?" Und bei jemandem, der sich schlecht fühlte, versuchte er, immer wieder zu fragen, was er schön fände und was ihm denn Spaß machen würde. Bis er genug davon zusammen hatte, um dem Benutzer zu sagen "O.K., heute geht es Ihnen nicht besonders gut. Sie haben Kopfweh. Und Sie müssen dauernd 'nein' sagen. Vielleicht hat Sie die Arbeit in den letzten Tagen zu sehr angestrengt. Aber denken Sie daran, wie schön ein Spaziergang im Park ist. Und wieviel Spaß Ihnen das Segeln macht. Außerdem dauern Kopfschmerzen nicht ewig."

Vielleicht merkst du schon etwas? Der Computer hat sich nicht wirklich mit dem Benutzer "unterhalten". Das kann er ja nicht. Er hat nur dessen Eingaben auf eine bestimmte Weise "sortiert" und ihm zurückgegeben.

Aber diese Art, in der der Computer seinen "Gesprächspartnern" die eigenen Informationen "widerspiegeln" konnte, hat damals vielen Leuten sehr gefallen. Und selbst jene, die ganz genau wußten, daß es nur ein Computerprogramm war, mit dem sie "sprachen", wollten sich von dem Computer erklären lassen, warum es ihnen "nicht besonders gutging", und wollten sich von ihm wieder in eine bessere Stimmung lotsen lassen.

Dies ging so weit, daß die Studenten sich heimlich in den Computerraum schlichen, um sich von ELIZA "beraten" zu lassen. Der Mann, der das Programm entwickelt hatte, Professor



Joseph Weizenbaum, hatte seine Entwicklung nämlich ELIZA genannt. Und nun fing ELIZA an, ihm selbst Kopfschmerzen zu machen, ja es wurde ihm beinahe unheimlich.

Solche Wirkungen erzielt ein Computerprogramm, weil es die Fähigkeit hat, alles zu "verstehen" und nichts zu vergessen. Aber wichtiger an diesem Programm ist seine Eigenschaft, Informationen, die es erhalten hat, in der genau passenden Form zurückzugeben.

Das kann entweder in sehr freundlicher oder auch unfreundlicher Form geschehen – sozusagen nach Lust und Laune des Programmierers. In Wirklichkeit geht es aber immer darum, daß der Computer den erhaltenen Input in der gewünschten Weise umformt.

Entwickeln wir doch einfach selber ein Programm, das dem Anwender so ein bißchen nach dem Mund redet.

Zuerst soll es, nach Eingabe der Monatszahl, den Namen des Monats ausgeben. Dazu fragt es den Benutzer:

"DER WIEVIELTE MONAT DES JAHRES IST JETZT?"

Der Computer soll "erkennen", welchen Monatsnamen man dafür verwendet. Versuche zunächst selbst ein Programm dafür zu schreiben, bevor du dich mit unserem Vorschlag beschäftigst.

In unserem Beispiel ist der Schutz gegen "unsinnige" Eingaben erst in Zeile 140 untergebracht. Das hat eigentlich nur mit der Zahlenfolge zu tun. Diese "Sicherung" muß, wie du siehst, nicht unbedingt direkt nach der INPUT-Zeile stehen.

Für den Monat 12, also Dezember, ist etwas Besonderes vorgesehen. Schließlich ist es der Weihnachtsmonat, und die Antworten des Computers müssen nicht unbedingt alle demselben Schema folgen (siehe Abbildung HC 31).

Eigentlich kann man die END-Befehle in den Antwortzeilen ebensogut weglassen. Die einzelnen Bedingungen hätten nämlich gar nichts anderes zugelassen als die richtigen Antworten. Doch manche Programmierer halten es für guten Stil, auch dann den END einzufügen, wenn das Programm ohnehin von selbst enden würde.

Klar, wenn es nichts mehr zu tun hat und auch nicht in eine andere Programmzeile zurückgeschickt wird, hält es an. Ärgerlich ist es allerdings, wenn man einen END-Befehl vergißt, der unbedingt erforderlich ist. Daher schadet es kaum, lieber einen END-Befehl mehr zu verwenden als nötig.

Bis jetzt übersetzt das Programm lediglich Zahlen in Begriffe. So etwas hatten wir zum Beispiel schon beim Programm, das die Namen der Wochentage ausgeben sollte. Nun soll der Computer das Ergebnis "kommentieren". Er soll behaupten, daß seiner Meinung nach der vom Benutzer gewählte Monat der schönste Monat im Jahr ist (siehe Abbildung HC 32).

Hierfür machen wir aus dem END ein GOTO und lassen das Programm in jener Zeile weiterarbeiten, in der der Computer folgende Meldung ausgibt:

"DAS IST DER SCHOENSTE MONAT IM JAHR."

Aber Vorsicht! Sobald du wie in diesem Fall eine Ausnahme einprogrammiert hast, mußt du sie auch weiterhin berücksichtigen. In unserem Beispiel hätte es wenig Sinn, von Zeile 130 auf die Zeile 150 zu springen wie bei den anderen Monaten. Deshalb sind für den Dezember einige besondere PRINT-Zeilen vorgesehen. Sie zeichnen einen Tannenbaum auf den Bildschirm.

Auch hier empfehlen wir, es erst selber zu probieren. Entweder behelfen wir uns dabei mit einer Vorzeichnung, also der Kästchen-Version des Bildschirms auf kariertem Papier, oder wir versuchen es gleich unmittelbar auf dem Monitor.

```
10 PRINT "WELCHEN MONAT IM JAHR HABEN WIR?"
20 INPUT "BITTE DIE ZAHL ";A
30 IF A<1 THEN PRINT "DAS VERSTEHE ICH NICHT!":GOTO 20
40 IF A=1 THEN PRINT "ACH, WIR HABEN JANUAR!":END
50 IF A=2 THEN PRINT "ACH, WIR HABEN FEBRUAR!":END
60 IF A=3 THEN PRINT "ACH, WIR HABEN MAERZ!":END
70 IF A=4 THEN PRINT "ACH, WIR HABEN APRIL!":END
80 IF A=5 THEN PRINT "ACH, WIR HABEN MAI!":END
90 IF A=6 THEN PRINT "ACH, WIR HABEN JUNI!":END
100 IF A=7 THEN PRINT "ACH, WIR HABEN JULI!":END
110 IF A=8 THEN PRINT "ACH, WIR HABEN AUGUST!":END
120 IF A=9 THEN PRINT "ACH, WIR HABEN SEPTEMBER!":END
130 IF A=10 THEN PRINT "ACH, WIR HABEN OKTOBER!":END
140 IF A=11 THEN PRINT "ACH, WIR HABEN NOVEMBER!":END
150 IF A=12 THEN PRINT "ACH, WEIHNACHTSZEIT!":END
160 IF A>12 THEN PRINT "EIN JAHR HAT, GLAUBE ICH, 12 MONATE,    ODER?":GOTO 10

READY.
```

Abb. HC 31

```
10 PRINT "WELCHEN MONAT IM JAHR HABEN WIR?"
20 INPUT "BITTE DIE ZAHL ";A
30 IF A<1 THEN PRINT "DAS VERSTEHE ICH NICHT!":GOTO 20
40 IF A=1 THEN PRINT "ACH, WIR HABEN JANUAR!":GOTO 170
50 IF A=2 THEN PRINT "ACH, WIR HABEN FEBRUAR!":GOTO 170
60 IF A=3 THEN PRINT "ACH, WIR HABEN MAERZ!":GOTO 170
70 IF A=4 THEN PRINT "ACH, WIR HABEN APRIL!":GOTO 170
80 IF A=5 THEN PRINT "ACH, WIR HABEN MAI!":GOTO 170
90 IF A=6 THEN PRINT "ACH, WIR HABEN JUNI!":GOTO 170
100 IF A=7 THEN PRINT "ACH, WIR HABEN JULI!":GOTO 170
110 IF A=8 THEN PRINT "ACH, WIR HABEN AUGUST!":GOTO 170
120 IF A=9 THEN PRINT "ACH, WIR HABEN SEPTEMBER!":GOTO 170
130 IF A=10 THEN PRINT "ACH, WIR HABEN OKTOBER!":GOTO 170
140 IF A=11 THEN PRINT "ACH, WIR HABEN NOVEMBER!":GOTO 170
150 IF A=12 THEN PRINT "ACH, WEIHNACHTSZEIT!":GOTO 190
160 IF A>12 THEN PRINT "EIN JAHR HAT, GLAUBE ICH, 12 MONATE,    ODER?":GOTO 10
170 PRINT:PRINT:PRINT "DAS IST DER SCHOENSTE MONAT IM JAHR."

READY.
```

Abb. HC 32

```
10 PRINT "WELCHEN MONAT IM JAHR HABEN WIR?"
20 INPUT "BITTE DIE ZAHL ";A
30 IF A<1 THEN PRINT "DAS VERSTEHE ICH NICHT!":GOTO 20
40 IF A=1 THEN PRINT "ACH, WIR HABEN JANUAR!":GOTO 170
50 IF A=2 THEN PRINT "ACH, WIR HABEN FEBRUAR!":GOTO 170
60 IF A=3 THEN PRINT "ACH, WIR HABEN MAERZ!":GOTO 170
70 IF A=4 THEN PRINT "ACH, WIR HABEN APRIL!":GOTO 170
80 IF A=5 THEN PRINT "ACH, WIR HABEN MAI!":GOTO 170
90 IF A=6 THEN PRINT "ACH, WIR HABEN JUNI!":GOTO 170
100 IF A=7 THEN PRINT "ACH, WIR HABEN JULI!":GOTO 170
110 IF A=8 THEN PRINT "ACH, WIR HABEN AUGUST!":GOTO 170
120 IF A=9 THEN PRINT "ACH, WIR HABEN SEPTEMBER!":GOTO 170
130 IF A=10 THEN PRINT "ACH, WIR HABEN OKTOBER!":GOTO 170
140 IF A=11 THEN PRINT "ACH, WIR HABEN NOVEMBER!":GOTO 170
150 IF A=12 THEN PRINT "ACH, WEIHNACHTSZEIT!":GOTO 190
160 IF A>12 THEN PRINT "EIN JAHR HAT, GLAUBE ICH, 12 MONATE,     ODER?":GOTO 10
170 PRINT:PRINT:PRINT "DAS IST DER SCHOENSTE MONAT IM JAHR."
180 GOTO 280
190 PRINT "      *
200 PRINT"     ***     FROEHLICHE
210 PRINT"    o * |
220 PRINT "    *****
230 PRINT "     @*+     WEIHNACHTEN!
240 PRINT"  ********.
250 PRINT" |   *   |
260 PRINT " **********
270 PRINT "     ***
```


Abb. HC 33

Das ganze Programm sieht mittlerweile so aus wie in Abbildung HC 33.

Macht bereits einen ganz passablen Eindruck, nicht wahr? Doch könnte man den Dialog noch ein wenig ausbauen. Warum lassen wir das Programm den Benutzer nicht mal fragen, ob er derselben Meinung ist oder ob er nicht einen anderen Monat sehr viel schöner findet?

Das bedeutet: aus dem END wird wieder ein GOTO. Dann setzen wir das Programm mit Hilfe einer INPUT-Zeile fort.

Das Programm soll nunmehr fragen, ob der Benutzer einen anderen Monat schöner findet. Der Benutzer gibt wiederum eine Zahl zwischen 1 und 12 ein. Auch hier müssen wir das Programm gegen falsche Eingaben sichern. Die dazu benötigte Programmzeile ähnelt der Zeile 140.

Diesmal soll das Programm nicht nur antworten:

"STIMMT, DAS IST DER SCHOENSTE MONAT IM JAHR."

Es soll vielmehr herausfinden, in welcher Jahreszeit der Lieblingsmonat des Benutzers liegt. Und dann soll es zustimmen, daß diese Jahreszeit die schönste ist. Für den Dezember wollen wir uns etwas Besonderes einfallen lassen.

Die Jahreszeiten teilen wir uns am besten folgendermaßen ein:

- Die Monate Dezember (12), Januar (1) und Februar (2) stehen für den Winter.
- Die Monate März (3), April (4) und Mai (5) gehören zum Frühling.
- Die Monate Juni (6), Juli (7) und August (8) sind der Sommer.
- Die Monate September (9), Oktober (10) und November (11) bleiben für den Herbst.

Sicher ist es dir schon aufgefallen: Etwas stimmt hier nicht ganz. Wenn du in den Kalender schaust, siehst du, daß diese Einteilung eigentlich ungenau ist. Die Jahreszeiten wechseln nämlich mitten in einem Monat. Wie du weißt, liegt im Jahr 1986 der Frühlingsanfang auf dem 20. März. Die Sommersonnenwende, also Sommeranfang, ist am 21. Juli, Herbstanfang haben wir am 22. September, und der Winteranfang ist kurz vor Weihnachten, am 22. Dezember.

Bei unserem Vorhaben reicht jedoch die etwas gröbere Aufteilung. Falls deine Ansprüche weiter gehen, spricht nichts dagegen, an dem Programm noch etwas zu tüfteln. Vielleicht

kannst du es später einmal so ausbauen, daß es genauere Antworten gibt. Warum solltest du es nicht dazu befähigen, das Datum auf den Tag genau zu verarbeiten? Dann würde es die Benutzer-Eingaben mit dem genauen Datum vergleichen, an dem die Jahreszeiten laut Kalender wechseln.

Doch zurück zu unserem Dialogprogramm. Nach der zweiten Eingabe wird sortiert. Die zweite Eingabe wollen wir einer eigenen Variablen zuordnen. Die Variable erhält den Variablen-Namen B. Wenn die Eingabe 12, 1 oder 2 ist, dann soll das Programm als Antwort ausgeben:

"STIMMT, DER WINTER IST AM SCHOENSTEN."

Mit Frühling, Herbst und Sommer machen wir es ebenso. Inzwischen dürfte das Programm so aussehen wie in Abbildung HC 34.

Für den Monat Dezember soll das Programm der Antwort ein bißchen "Sahne aufsetzen". Hierzu entwerfen wir eine eigene "Routine". So nennt man einen Programmteil, der eine besondere Aufgabe hat. Als Gag soll nämlich unser Programm den Monat, in dem der Benutzer sich befindet (gemeint ist der zuerst eingegebene Monat), mit dem Dezember "vergleichen". Daraufhin gibt es auf dem Bildschirm an, wie viele Monate es noch bis zum Dezember dauert. Der Computer muß hierfür lediglich die zuerst eingegebene Variable A von 12 subtrahieren. Er rechnet also "12 minus A". Für den Output schreiben wir dann folgende PRINT-Zeile:

```
PRINT "ES DAUERT NOCH ";12-A;" MONATE BIS DEZEMBER."
```

Doch tauchen dabei zwei Schwierigkeiten auf. Die erste hat etwas mit der Sprache zu tun, die *wir* sprechen – genauer: mit unserer Grammatik. Sobald nämlich 12-A die Zahl 1 ergibt, muß auf dem Bildschirm statt der Mehrzahl MONATE die Einzahl MONAT erscheinen. Für diesen Fall benötigen wir eine zusätzliche Zeile.

Zwar könnte man in solch einem Fall die Mehrzahlendung in Klammern setzen, was folgendermaßen aussähe:

```
PRINT "ES DAUERT NOCH ";12-A;" MONAT(E) BIS DEZEMBER."
```

Leider klappt das nicht bei allen Wörtern. Zum Beispiel wird die Mehrzahl von *Topf* nicht allein durch ein Endungs-"e" gebildet, sondern innerhalb des Wortes ändert sich auch ein Vokal

```
10 PRINT "WELCHEN MONAT IM JAHR HABEN WIR?"
20 INPUT "BITTE DIE ZAHL ";A
30 IF A<1 THEN PRINT "DAS VERSTEHE ICH NICHT!":GOTO 20
40 IF A=1 THEN PRINT "ACH, WIR HABEN JANUAR!":GOTO 170
50 IF A=2 THEN PRINT "ACH, WIR HABEN FEBRUAR!":GOTO 170
60 IF A=3 THEN PRINT "ACH, WIR HABEN MAERZ!":GOTO 170
70 IF A=4 THEN PRINT "ACH, WIR HABEN APRIL!":GOTO 170
80 IF A=5 THEN PRINT "ACH, WIR HABEN MAI!":GOTO 170
90 IF A=6 THEN PRINT "ACH, WIR HABEN JUNI!":GOTO 170
100 IF A=7 THEN PRINT "ACH, WIR HABEN JULI!":GOTO 170
110 IF A=8 THEN PRINT "ACH, WIR HABEN AUGUST!":GOTO 170
120 IF A=9 THEN PRINT "ACH, WIR HABEN SEPTEMBER!":GOTO 170
130 IF A=10 THEN PRINT "ACH, WIR HABEN OKTOBER!":GOTO 170
140 IF A=11 THEN PRINT "ACH, WIR HABEN NOVEMBER!":GOTO 170
150 IF A=12 THEN PRINT "ACH, WEIHNACHTSZEIT!":GOTO 190
160 IF A>12 THEN PRINT "EIN JAHR HAT, GLAUBE ICH, 12 MONATE,    ODER?":GOTO 10
170 PRINT:PRINT:PRINT "DAS IST DER SCHOENSTE MONAT IM JAHR."
180 GOTO 280
190 PRINT "      *
200 PRINT"    ***    FROEHLICHE
210 PRINT"   o * |
220 PRINT "   *****
230 PRINT "    ●*♠    WEIHNACHTEN!
240 PRINT"  *******
250 PRINT" |   *   |
260 PRINT " *********
270 PRINT "    ***
280 INPUT "ODER MAGST DU EINEN ANDEREN MONAT       LIEBER? DIE ZAHL ";B
290 IF B=12 OR B=1 OR B=2 THEN PRINT "STIMMT. DER WINTER IST AM SCHOENSTEN."
300 IF B=3 OR B=4 OR B=5 THEN PRINT"STIMMT. DER FRUEHLING IST AM SCHOENSTEN."
310 IF B=6 OR B=7 OR B=8 THEN PRINT "STIMMT. DER SOMMER IST AM SCHOENSTEN."
320 IF B=9 OR B=10 OR B=11 THEN PRINT"STIMMT. DER HERBST IST AM SCHOENSTEN."

READY.
```

Abb. HC 34

(Töpfe). Außerdem erinnert eine solche angehängte Klammer an Bürokratendeutsch. Uns ist das jedenfalls nicht elegant genug. Wir meinen, man sollte den Computer so perfekt wie möglich "reden" lassen. Der Commodore 64 schränkt uns ohnehin oft genug ein. Zwar bietet er reichlich freien Speicherplatz. Aber eine BASIC-Programmzeile darf bei ihm einschließlich Zeilennummer nicht länger sein als *zwei* Bildschirmzeilen, also 80 Zeichen.

Die zweite Schwierigkeit entsteht, sobald der eingegebene Monat Dezember heißt. Für diesen Fall, nämlich für

A = 12

brauchen wir ebenfalls eine besondere Programmzeile. Denn die Meldung

"ES DAUERT NOCH 0 MONATE BIS DEZEMBER."

klingt etwas eigenartig. Wir sollten unser Programm zu etwas "intelligenteren" Aussagen veranlassen. Warum gestalten wir die Antwort nicht so, als habe der Computer nicht nur von selber "erkannt", daß beispielsweise gerade Dezember ist, sondern obendrein vermutet, daß der Dezember der Lieblingsmonat des Benutzers ist? Das erreichen wir durch folgende Programmzeile:

```
IF A=12 AND B=12 THEN PRINT "WIE SCHOEN!

ES IST DEIN LIEBLINGSMONAT

DEZEMBER.":END
```

Nun fügen wir noch eine Abfrage hinzu. Das Programm fordert den Benutzer auf einzugeben, der wievielte Dezember es ist. Daraufhin vergleicht es dieses Datum mit dem 24. Dezember und gibt verschiedene Antworten darauf. Diese Antworten richten sich danach, ob es noch eine, zwei oder drei Wochen bis Weihnachten sind oder ob Weihnachten gar schon vorbei ist.

Die endgültige Fassung des fertigen Programms könnte jetzt etwa so aussehen wie Abbildung HC 35.

Nach dem gleichen Strickmuster ließe sich auch ein Programm schreiben, das dem Benutzer ausrechnet, wie viele Monate und schließlich wie viele Tage noch bis zu seinem Geburtstag vergehen.

```
10 PRINT "WELCHEN MONAT IM JAHR HABEN WIR?"
20 INPUT "BITTE DIE ZAHL ";A
30 IF A<1 THEN PRINT "DAS VERSTEHE ICH NICHT!":GOTO 20
40 IF A=1 THEN PRINT "ACH, WIR HABEN JANUAR!":GOTO 170
50 IF A=2 THEN PRINT "ACH, WIR HABEN FEBRUAR!":GOTO 170
60 IF A=3 THEN PRINT "ACH, WIR HABEN MAERZ!":GOTO 170
70 IF A=4 THEN PRINT "ACH, WIR HABEN APRIL!":GOTO 170
80 IF A=5 THEN PRINT "ACH, WIR HABEN MAI!":GOTO 170
90 IF A=6 THEN PRINT "ACH, WIR HABEN JUNI!":GOTO 170
100 IF A=7 THEN PRINT "ACH, WIR HABEN JULI!":GOTO 170
110 IF A=8 THEN PRINT "ACH, WIR HABEN AUGUST!":GOTO 170
120 IF A=9 THEN PRINT "ACH, WIR HABEN SEPTEMBER!":GOTO 170
130 IF A=10 THEN PRINT "ACH, WIR HABEN OKTOBER!":GOTO 170
140 IF A=11 THEN PRINT "ACH, WIR HABEN NOVEMBER!":GOTO 170
150 IF A=12 THEN PRINT "ACH, WEIHNACHTSZEIT!":GOTO 190
160 IF A>12 THEN PRINT "EIN JAHR HAT, GLAUBE ICH, 12 MONATE,    ODER?":GOTO 10
170 PRINT:PRINT:PRINT "DAS IST DER SCHOENSTE MONAT IM JAHR."
180 GOTO 280
190 PRINT"      *
200 PRINT"     ***     FROEHLICHE
210 PRINT"    o * |
220 PRINT"    *****
230 PRINT"     ***     WEIHNACHTEN!
240 PRINT"   *******
250 PRINT"  |   *   |
260 PRINT" *********
270 PRINT"     ***
280 INPUT "ODER MAGST DU EINEN ANDEREN MONAT        LIEBER? DIE ZAHL ";B
290 IF B=12 OR B=1 OR B=2 THEN PRINT "STIMMT. DER WINTER IST AM SCHOENSTEN."
```

```
300 IF B=3 OR B=4 OR B=5 THEN PRINT"STIMMT. DER FRUEHLING IST AM SCHOENSTEN."
310 IF B=6 OR B=7 OR B=8 THEN PRINT "STIMMT. DER SOMMER IST AM SCHOENSTEN."
320 IF B=9 OR B=10 OR B=11 THEN PRINT"STIMMT. DER HERBST IST AM SCHOENSTEN."
330 IF B=12 THEN GOTO 400
340 IF 12-B=1 THEN PRINT "ES DAUERT NOCH EINEN MONAT BIS DEZEMBER.":END
350 PRINT "ES DAUERT NOCH ";12-B;" MONATE BIS DEZEMBER.":END
400 INPUT "DER WIEVIELTE DEZEMBER IST HEUTE";C
410 IF C<10 THEN PRINT "NOCH EIN BISSCHEN GEDULD.":END
420 IF C<20 THEN PRINT "JETZT WIRD ES SCHON LANGSAM SPANNEND.":END
430 IF C<25 THEN PRINT "EIN SCHOENES FEST WUENSCHT AUCH DEIN    COMPUTER.":END
440 PRINT "HOFFENTLICH WAR WEIHNACHTEN SCHOEN!":END

READY.
```

Abb. HC 35

Man könnte sich im Rahmen eines so aufgebauten Programms mit dem Computer auch darüber "unterhalten", wie lange es noch bis zu den großen Ferien dauert und was man dann unternehmen will. Vielleicht gestaltest du sogar das Programm so trickreich, daß der Computer dem Benutzer ein paar Vorschläge dazu macht.

Wer es leid ist, immer an den Handknöcheln abzuzählen, wie viele Tage ein bestimmter Monat hat, kann statt dessen ein kleines Programm dafür schreiben.

Programmieren mit Plan

Im letzten Kapitel haben wir schon recht umfangreiche Programme geschrieben. Auf den ersten Blick sehen die Listings ziemlich kompliziert aus. Dennoch dürfte es dir nicht allzu schwer gefallen sein, die Programme aufzubauen. Wir hatten uns nämlich vorher die einzelnen Schritte genau überlegt. Und das Programm wurde von uns zunächst in einzelne Bausteine zerlegt. Diese haben wir schließlich Stück für Stück zusammengefügt.

Angesichts dessen, was wir bisher gelernt haben, sollten wir uns einmal fragen: Was ist Programmieren eigentlich? Wie du mittlerweile festgestellt hast, auf jeden Fall keine Geheimwissenschaft, sondern im Grunde eine reichlich simple Sache. Wesentlich sind die folgenden drei Regeln:

Indem wir ein Programm schreiben, "sagen" wir dem Computer nur,

1. was genau der Computer tun soll,
2. in welchen Schritten er es tun soll, und
3. in welcher Reihenfolge er es tun soll.

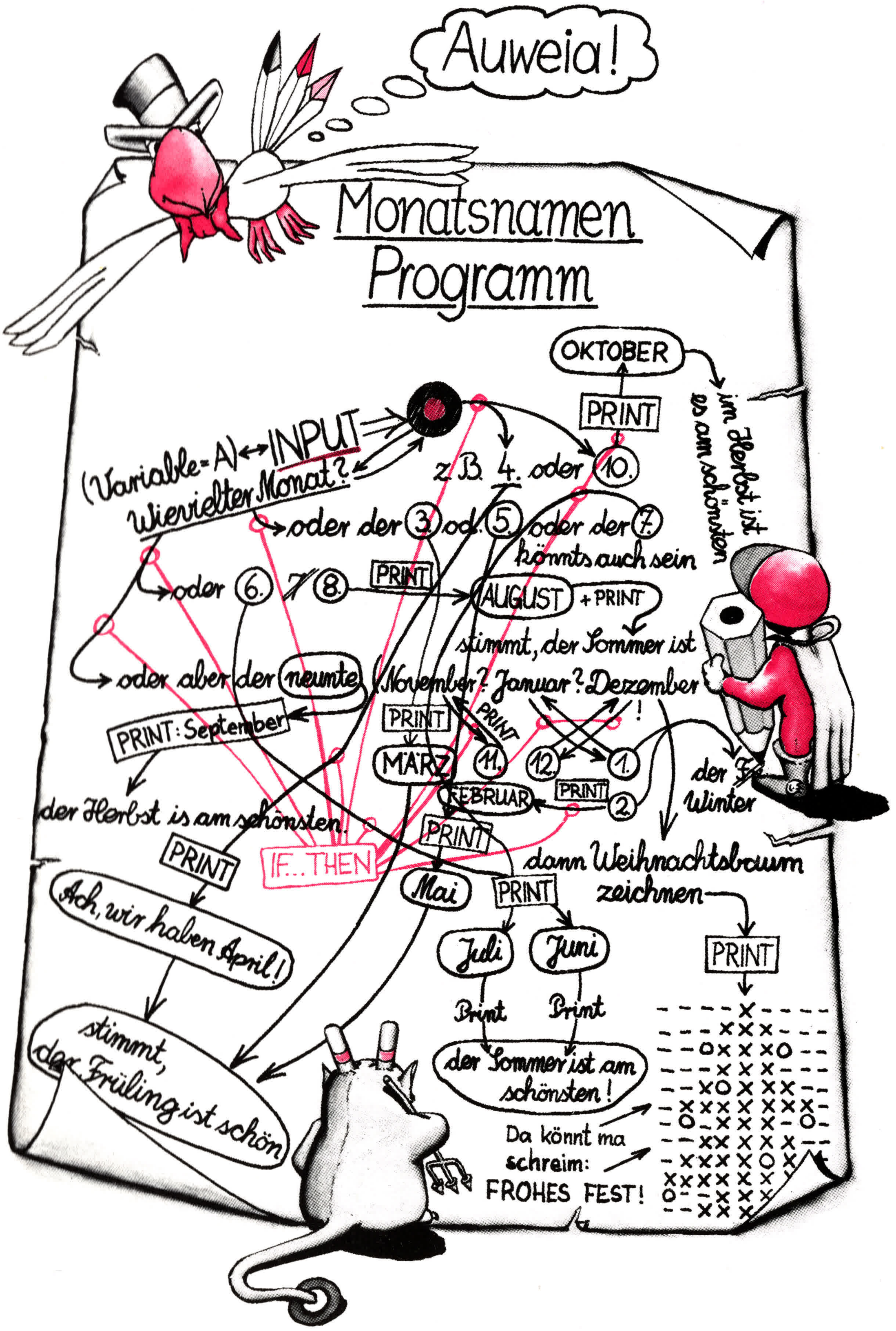
Woraus setzen Programme sich zusammen? Programme bestehen aus Eingaben und Ausgaben, Befehlen, Anweisungen und "Entscheidungen".

Nehmen wir an, du hast eine Programm-Idee. Du weißt schon ungefähr, welche Befehle und sonstigen BASIC-Ausdrücke du dazu verwenden willst.

Bevor du nun das Programm in BASIC schreibst, bietet es sich an, einen Plan zu machen. Das erleichtert die Übersicht. Allerdings sollten wir dabei anders vorgehen als unsere Freunde.

Die Computerprofis haben sich für diesen Zweck etwas besonders Nützliches ausgedacht, nämlich das "Flußdiagramm". Der englische Begriff dafür lautet "flow chart" (gesprochen "floh tschahrt"). Dieser Fachausdruck wird auch bei uns häufig verwendet, vor allem von EDV-Leuten.

Mit Diagrammen hast du wahrscheinlich schon in der Schule zu tun gehabt. Man begegnet ihnen meist im Erdkundeunter-


Auweia!
Monatsnamen Programm
(Variable=A)↔INPUT
Wievielter Monat?
z.B. 4. oder 10.
oder der 3. od. 5. oder der 7.
könnts auch sein
oder 6. 7 8.
PRINT
AUGUST + PRINT
stimmt, der Sommer ist
oder aber der neunte
November? Januar? Dezember!
PRINT: September
der Herbst is am schönsten.
im Herbst ist es am schönsten
MÄRZ
11.
12.
1.
2
FEBRUAR
der Winter
IF...THEN
dann Weihnachtsbaum zeichnen
Mai
Juli
Juni
Print
Ach, wir haben April!
stimmt, der Frühling ist schön
der Sommer ist am schönsten!
Da könnt ma schreim: FROHES FEST!

richt oder in den naturwissenschaftlichen Fächern, also Physik, Chemie oder Biologie – auch in Mathematik.

In einem Diagramm werden meist Zahlenzusammenhänge zeichnerisch dargestellt. Mit Hilfe eines Flußdiagramms zeichnen wir auf, wie Daten oder Informationen zusammenhängen und wie bzw. wohin sie vom Programm bewegt werden.

Eigentlich kann man sich das ganz gut vorstellen, wie die Daten "fließen" – von der Eingabe über die verschiedenen Verarbeitungsschritte bis zur Ausgabe.

Statt des Ausdrucks Flußdiagramm verwendet man auch oft die Bezeichnung "Programmablaufplan" – ein schon fast unhandliches Wortungetüm. Allerdings drückt dieser Fachausdruck sehr genau aus, worum es geht.

Anhand eines solchen Plans legen wir jedenfalls fest, in welchen Schritten das Programm abläuft. Mit Hilfe von einfachen grafischen Symbolen zeigen wir, welche "Logik" in einem Programm steckt.

Vorhin haben wir das Programm in Worten geschildert. Nun setzen wir diese Programmbeschreibung in einen Plan um. Der veranschaulicht den Programmablauf im Computer deutlicher als unsere vorherige Programmbeschreibung. Die "Computer-Logik" tritt jetzt klar hervor.

Für jede Art von Information und Verarbeitung innerhalb eines Programmes ist im Flußdiagramm ein bestimmtes Symbol vorgesehen. Diese "Sinnbilder" sind oft so einfach, daß sie sich auf den ersten Blick verstehen lassen. Sie bestehen aus Kreisen, Rechtecken, Rauten, Ovalen und einigen anderen Formen, die im Augenblick für uns noch nicht so wichtig sind.

Verbunden werden diese Symbole mit einer Linie. Und auf dieser Linie zeigen kleine Pfeile an, in welcher Richtung das Programm abläuft – man könnte auch sagen: in welche Richtung die Daten "fließen". Auf diese Weise veranschaulichen wir den Ablauf eines Programms. Mit Hilfe des Flußdiagramms vermeidest du, daß zum Beispiel bei einer bedingten Verzweigung wichtige Schritte übersehen oder ganz einfach vergessen werden.

Stell dir vor, du möchtest dir ein Flugzeugmodell bauen. Dabei gehst du selbstverständlich (!) nach einem Plan vor. Dieser Plan ist entweder schon im Bausatz enthalten, oder du zeichnest ihn dir selbst. Nicht anders ist es, wenn man sich einen Pullover strickt: Dafür gibt es Strickmuster.

Beim Programmieren tun wir dasselbe: Wir machen uns vorher einen Plan. Oder wir halten uns an einen Plan, der uns vielleicht schon vorliegt. Ohne gute Planung kein zufriedenstellendes Ergebnis.

Allerdings kann es auch vorkommen, daß man bei der Durchführung den Plan selbst noch mal ändern muß. Möglicherweise hast auch du das beim Basteln oder Stricken schon erlebt.

Ähnlich ist es auch beim Programmieren. Sobald es daran geht, das Flußdiagramm in die entsprechenden BASIC-Befehle umzusetzen, entdeckt man zuweilen, daß es nicht ganz so geht, wie es einem zunächst vorgeschwebt hat. Da findet sich eine andere, bessere Lösung, indem man ein bißchen herumprobiert. Und es wäre sinnlos, sich sklavisch an den Programmablauf zu halten – nur weil er eben da ist. Statt dessen ändert man nachträglich den Plan oder erweitert ihn.

Das Flußdiagramm soll helfen, komplizierte Programmabläufe übersichtlich darzustellen. So behalten wir die "Logik" unseres Programmaufbaus im Auge.

Mit einem Flußdiagramm können wir nicht nur verdeutlichen, was der Computer tun soll. Auch alltägliche Tätigkeiten, wie Einkaufen, zur Schule fahren, Kochen, Fahrradputzen oder Hausaufgaben erledigen, haben einen logischen Ablauf. Und diese "Logik" läßt sich ebenfalls in Form von den Flußdiagrammen darlegen.

Am Beispiel solcher Alltagsverrichtungen kann man vorzüglich üben, wie ein Programmablaufplan aufgebaut sein muß. Dazu nun einige Beispiele:

Auf jeden Fall muß das Programm einen unmißverständlich festgelegten Anfang und ein Ende haben. Den Beginn und den Schluß des Programms stellen wir mit dem folgenden Symbol dar:

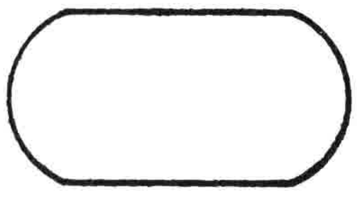


Abb. Z 2

Das nächste Sinnbild bedeutet, daß innerhalb eines Programms etwas geschehen soll (zum Beispiel etwas rechnen oder auf dem Bildschirm zeichnen, usw.) Es ist das Symbol für "Verarbeitung".

Abb. Z 3

Mit Hilfe dieser beiden Symbole kann man bereits einfache Flußdiagramme anlegen. Die Sinnbilder unseres Flußdiagramms mußt du lediglich mit Strichen oder Pfeilen verbinden.

Zuerst zeichnen wir einige Programmablaufpläne für einfache Tätigkeiten. Sie sind zwar im Buch bereits abgebildet, jedoch empfiehlt es sich, diese Flußdiagramme auf einem Block nachzuzeichnen. Das mag dir zunächst sehr simpel und überflüssig vorkommen. Doch gibt es einem die nötige Übung, wenn es später darum geht, größere und kompliziertere Diagramme zu zeichnen. Kariertes Papier ist dafür am günstigsten.

Abbildung Z 4 zeigt eine geradezu "primitive" Tätigkeit, nämlich: von einem Stuhl aufstehen, auf dem man gerade sitzt und sich anschließend wieder auf diesen Stuhl setzen.

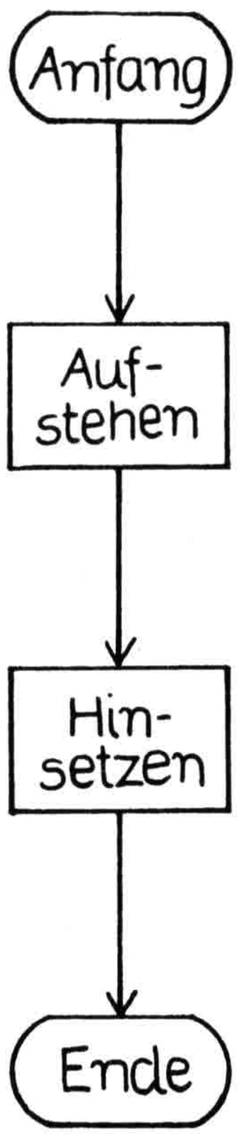


Abb. Z 4

Überlege dir jetzt selber einige möglichst einfache Tätigkeiten, und versuche sie als Programmablaufpläne aufzuschreiben. Dabei wirst du wahrscheinlich entdecken, daß es verschiedene Möglichkeiten gibt, ein und denselben Vorgang darzustellen.

Jemand nimmt einen Bleistift in die Hand. Wenn du dir diesen Vorgang kurz vorstellst, wird dir klar, daß sich eine so scheinbar einfache Handlung in unterschiedlich viele Einzelschritte zerlegen läßt.

Am leichtesten macht man es sich, wenn "Bleistift in die Hand nehmen" bereits als ein einziger Schritt gelten soll. Entsprechend kurz wird das Flußdiagramm (siehe Abbildung Z 5).

Abb. Z 5

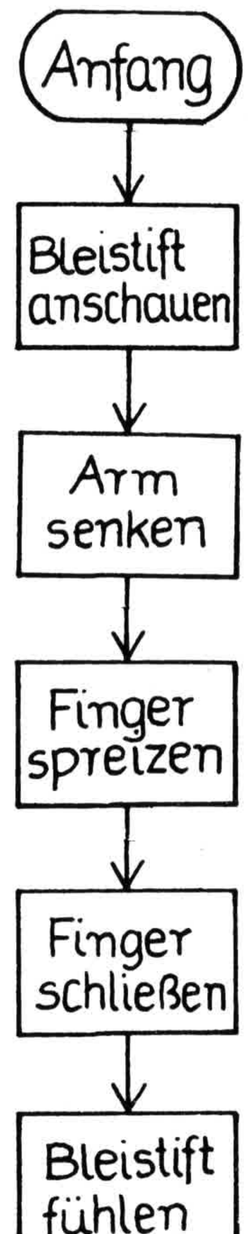


Abb. Z 6

Oder du untergliederst diese Tätigkeit in sehr viele Schritte:

- Zuerst den Arm nach dem Bleistift ausstrecken,
- dann die Hand senken,
- anschließend die Finger spreizen,
- schließlich den Bleistift fassen,
- sodann die Hand wieder heben,

(und so weiter).

Der dazugehörige Ablaufplan könnte etwa so aussehen, wie in Abbildung Z 6.

Ein drittes Symbol, das wir in Flußdiagrammen verwenden, ist die Raute. Wie in Abbildung Z 7 zu sehen, steht sie auf der stumpfen Ecke.

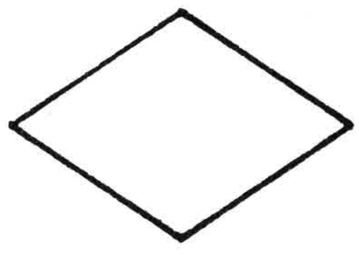


Abb. Z 7

Dieses Sinnbild bedeutet: Hier ist eine Entscheidung fällig. Das heißt, es wird eine Frage gestellt (Hunger?). Je nachdem, wie die Antwort ausfällt (Ja oder Nein), nimmt der Programmablauf eine bestimmte Richtung (essen oder nicht essen). Dieser Programmschritt ist bereits im vorherigen Band etliche Male vorgekommen. Er heißt "bedingte Verzweigung".

Stellen wir uns vor, jemand steht auf, geht zur Türe und öffnet sie. Dann geht er wieder genau an die Stelle zurück, von der er sich erhoben hat. Das klingt wirklich nicht sonderlich kompliziert – ist es eigentlich auch nicht. Und dennoch läßt sich eine solche Handlung nicht nur in mehrere Einzelschritte unterteilen, sondern es sind bereits einige Entscheidungen zu treffen. Wie könnte das entsprechende Flußdiagramm aussehen?

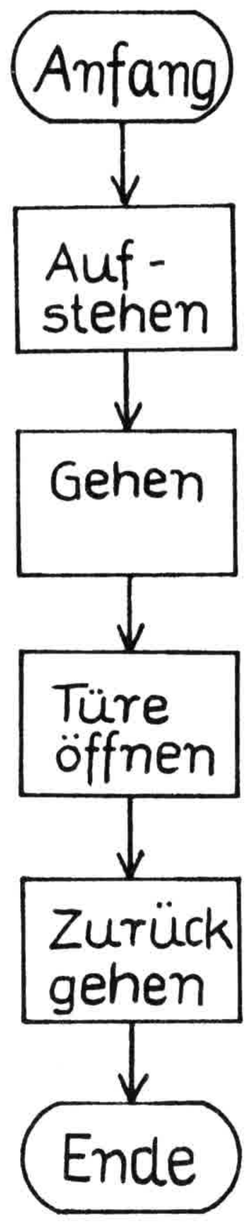


Abb. Z 8

In Abbildung Z 8 haben wir es uns nicht gerade sehr schwergemacht. Es kommen auch keine Entscheidungen vor. Und das bedeutet: Wir haben nicht so genau hingesehen. Versuche dir genau zu vergegenwärtigen, was beim Aufstehen, Hingehen, Türaufmachen und Zurückgehen alles geschieht. Du erkennst rasch, daß in so einem Vorgang viel mehr Einzeltätigkeiten enthalten sind, als man zunächst vermutet. Obendrein muß man dabei etliche Entscheidungen vornehmen (siehe Abbildung Z 9).

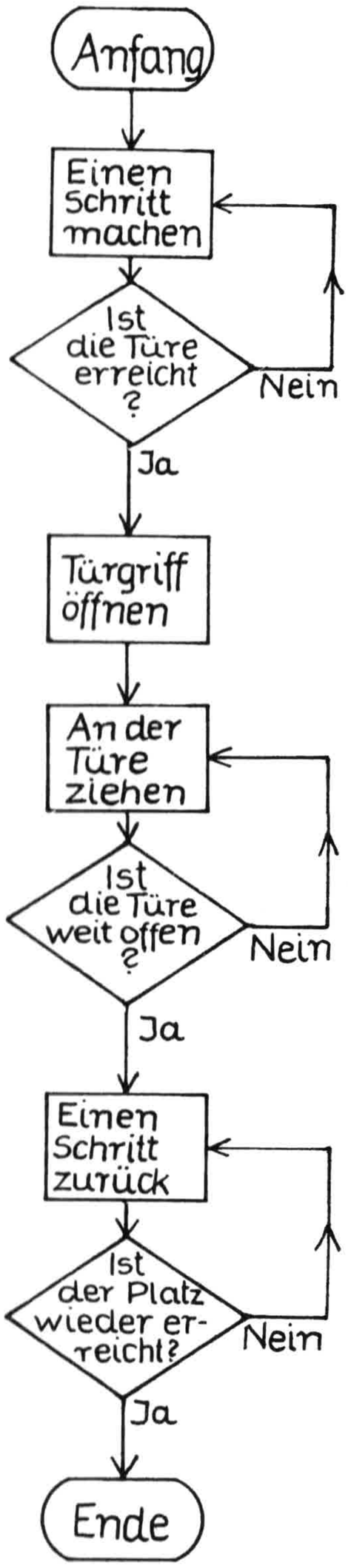


Abb. Z 9

Vielleicht sind dir das noch nicht genügend Teilschritte. Zweifellos könntest du in diesem Flußdiagramm auch noch all das einfügen, was im Gehirn, in den Nervenbahnen, in den Muskeln und Sehnen und Gelenken vor sich geht – und sämtliche Augenbewegungen. Dazu bräuchten wir allerdings ein Stück Papier, das etwa so groß wäre wie der Bodensee.

Wahrscheinlich ist es doch günstiger, als nächstes erst einmal den eigenen Schulweg als Flußdiagramm aufzuzeichnen. Wie das aussehen könnte, siehst du in Abbildung Z 10.

An einem Entscheidungspunkt kann sich das Programm in mehrere Wege verzweigen. Ebenso können mehrere Verzweigungen in eine Tätigkeit münden. Ein Beispiel dazu ist der Vorgang "Schulranzen packen" (siehe Abbildung Z 11).

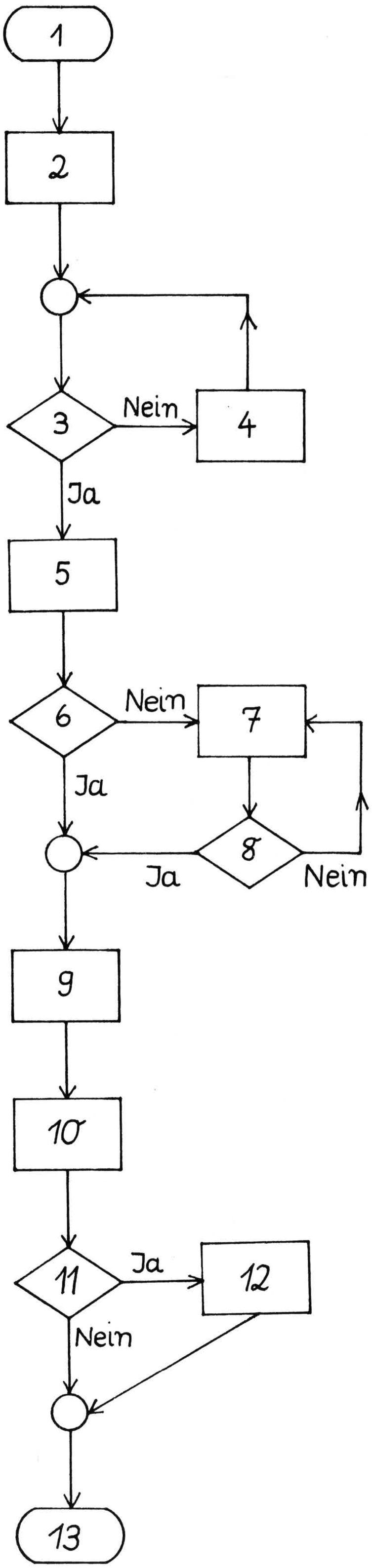


Abb. Z 10

1 Losgehen

2 100 m Kantstraße

3 Ist die Ampel auf grün ?

4 Warten

5 Meisterstr. 500 m, rechts abbiegen zur Parkstraße

6 Ist es schon 3/4 8 ?

7 Spielzeuggeschäft Auslagen ansehen

8 Ist es schon 5 vor 8 ?

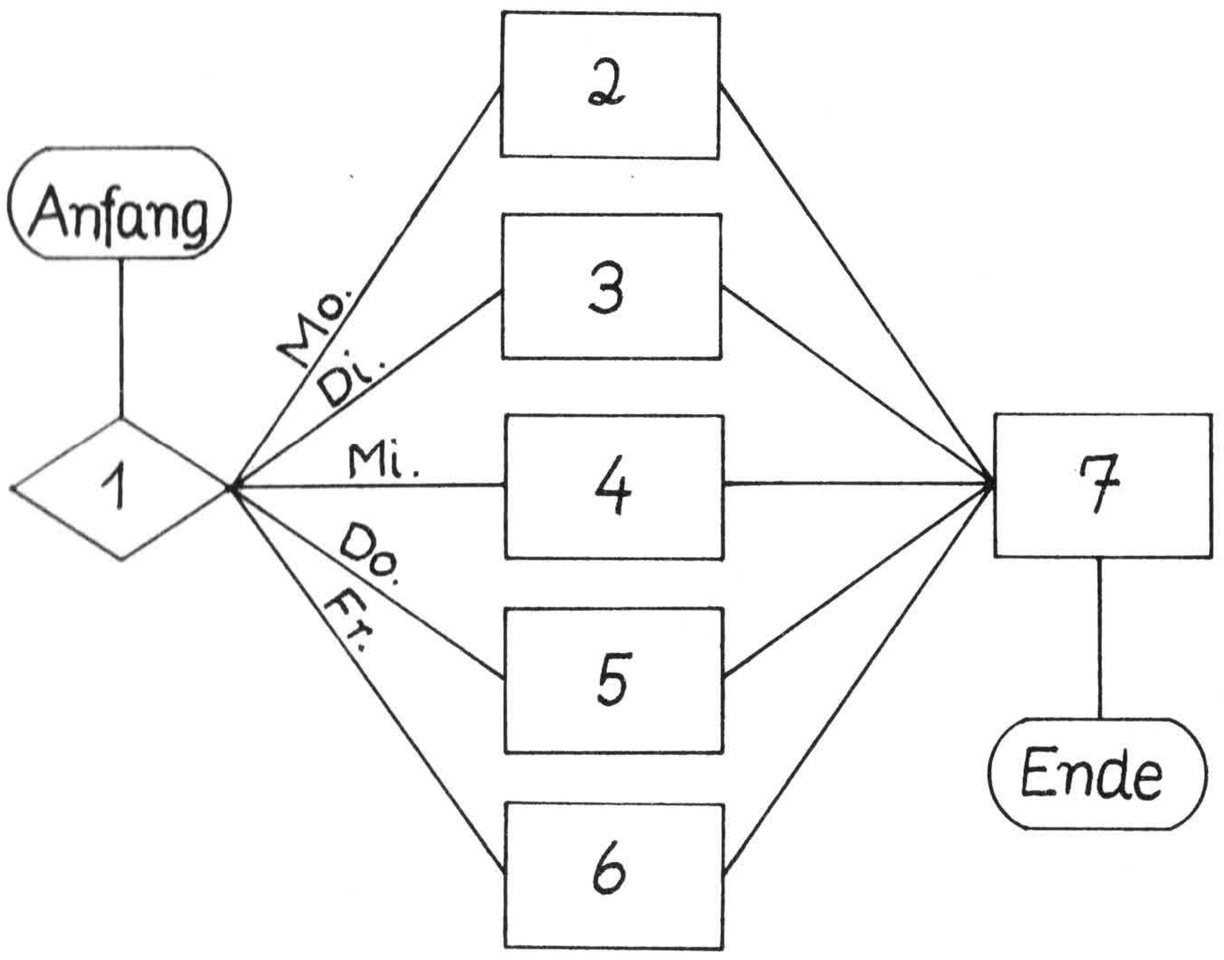
9 Bachstraße hinunterlaufen

10 In die Klasse gehen

11 Zu Spät ?

12 Ausrede erfinden,

13 Hinsetzen : Ende



1 Welcher Tag ist heute ?

2 Montag Packen : Deutsch
Rechnen
Handarbeit

3 Dienstag Packen : Rechnen
Heimat-u. Sachkunde
Religion
Deutsch

4 Mittwoch Packen : Deutsch
Rechnen
Förderunterricht
Schwimmen

5 Donnerstag Pck. : Deutsch
Rechnen
Religion
Turnen

6 Freitag Packen : Deutsch
Rechnen
Musik
Heimat-u. Sachkunde

7 Pausebrot einpacken
Schulranzen zumachen.

Abb. Z 11

Im Flußdiagramm "Schulweg" siehst du, daß man Stellen, an denen sich mehrere Zweige des Diagramms treffen, mit einem kleinen Kreis kennzeichnet.

Abb. Z 12

Ein wichtiges Symbol müssen wir noch erklären. Dann kannst du mit Hilfe der Flußdiagramme bereits ausgesprochen raffinierte BASIC-Programme entwerfen. Ein wenig erinnert das nun folgende Sinnbild an eine Raute. Wie in Abbildung Z 13 deutlich erkennbar, steht es jedoch im Unterschied zur Raute auf seiner Längsseite und nicht auf der stumpfen Ecke. In der Geometrie nennt man eine solche Figur "Parallelogramm".

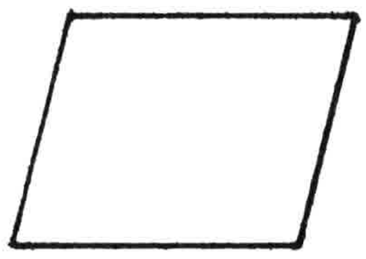


Abb. Z 13

Im Ablaufplan taucht dieses Parallelogramm an jenen Stellen des Programms auf, an denen vom Benutzer etwas erwartet wird – zum Beispiel eine Eingabe über die Tastatur. Also bedeutet das Sinnbild in diesem Fall "Input". Aber an den Stellen des Programms, an denen wir als Benutzer vom Computer etwas erwarten, benutzen wir dieses Symbol nicht. Den Output, etwa eine Bildschirmmeldung wie beispielsweise die Lösung einer Rechenoperation, stellen wir mit Hilfe des Rechtecks dar.

Es gibt noch einige andere Symbole in Flußdiagrammen. Allerdings kommen sie nicht sehr häufig vor. Meist haben sie etwas mit Peripheriegeräten zu tun, also beispielsweise mit Druckern, Plottern, Modems und ähnlichen Vorrichtungen.

Wie schon gesagt: Ein anderer Ausdruck für "Symbol" ist das Wort "Sinnbild". Dieser Begriff wird auch in der offiziellen "Deutschen Industrie-Norm" verwendet. Die Symbole für Programmablaufpläne sind in Deutschland nämlich genormt. Es gibt somit eine verbindliche Vorschrift dafür, welches Sinnbild welche Bedeutung hat. So kann jeder Programmierer die Flußdiagramme seiner Kollegen eindeutig lesen.

Leider sind diese Sinnbilder nicht weltweit gültig. In Amerika und Japan benützt man einige andere Symbole. Die bundesdeutschen Normen sind 1983 neu festgelegt worden. Auf diese Weise wollte man einem gemeinsamen internationalen Standard näherkommen. Seither verwendet man in der Bundesrepublik Deutschland offiziell für Input und Output nicht mehr das Parallelogramm, sondern ebenfalls das Rechteck. Doch hat sich diese neue Norm in der Praxis noch nicht durchgesetzt. Das sieht man auch immer wieder an Flußdiagrammen in Computerzeitschriften. Meistens finden wir dort nach wie vor das Parallelogramm als Symbol für INPUT/OUTPUT.

Für den Einsteiger ist die alte Norm eigentlich übersichtlicher. Deshalb bleiben wir auch in diesem Buch vorerst dabei. Denn für uns sollen ein Flußdiagramm und seine Sinnbilder vor allem ein Mittel sein, um ein Programm zu verstehen und selber zu entwerfen.

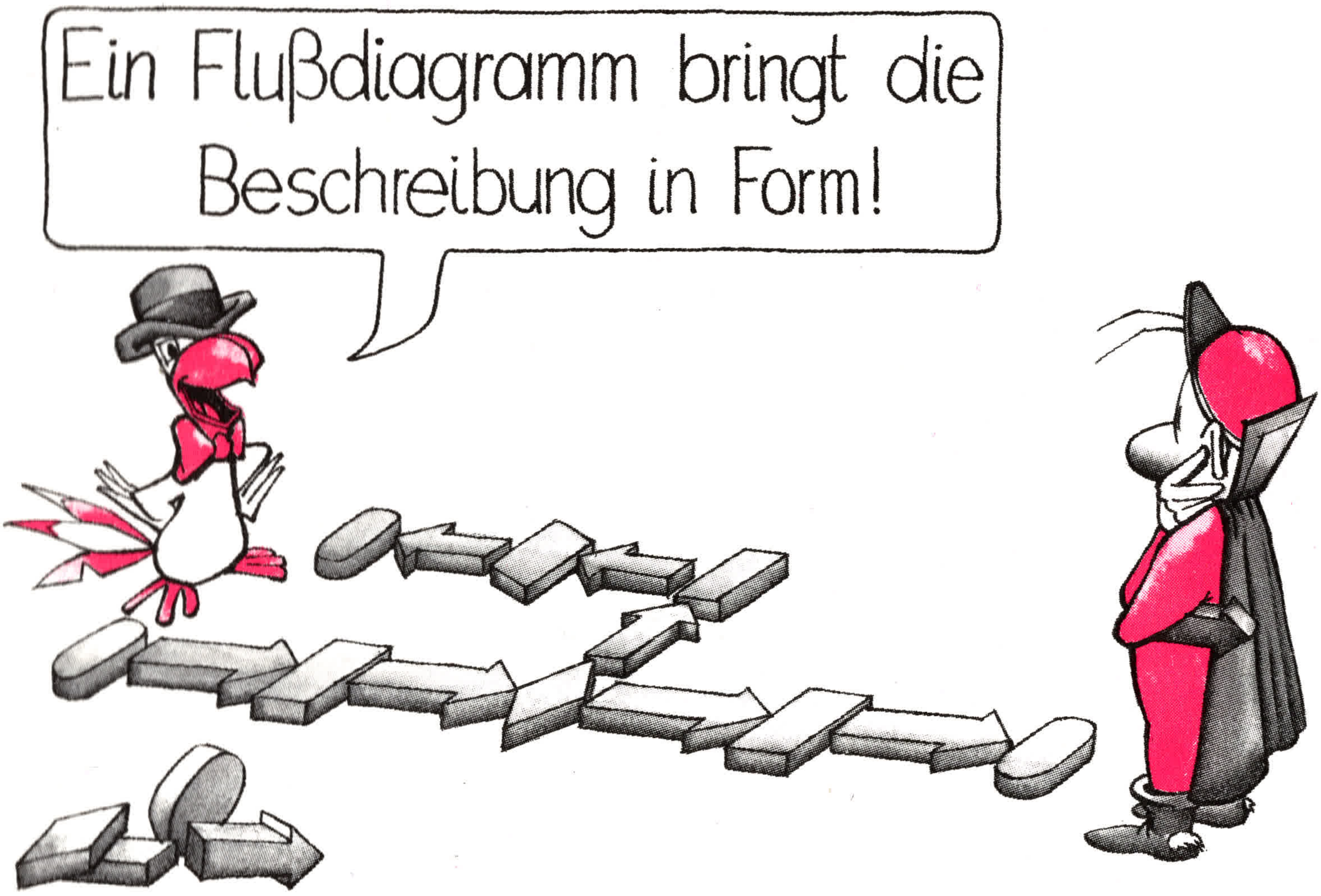
Eine Aufstellung der hierzulande derzeit gültigen Symbole haben wir bereits im Anhang des zweiten Bandes von "Computer kinderleicht" ("Lernen, Spielen, Programmieren") abgedruckt.

Jetzt wollen wir ein BASIC-Programm in den Programmablaufplan "übersetzen". Normalerweise gehst du zwar umgekehrt vor. Doch kann man mit Hilfe eines Flußdiagramms nicht nur ein Programm leichter entwerfen. Es hilft uns überdies, ein fertiges BASIC-Programm leichter zu durchschauen. Und es unterstützt uns beim Erlernen der "Computer-Logik".

Betrachten wir das einfache Programm von Seite 13. In BASIC geschrieben sieht es aus wie in Abbildung HC 4.

```
10 PRINT "ICH NENNE DIE ZAHL"
20 PRINT "NACH DEM FRAGEZEICHEN"
30 PRINT "MAL FUENF"
40 INPUT A
50 PRINT A*5
```

Abb. HC 4



Als Flußdiagramm sieht es aus wie in Abbildung Z 14.

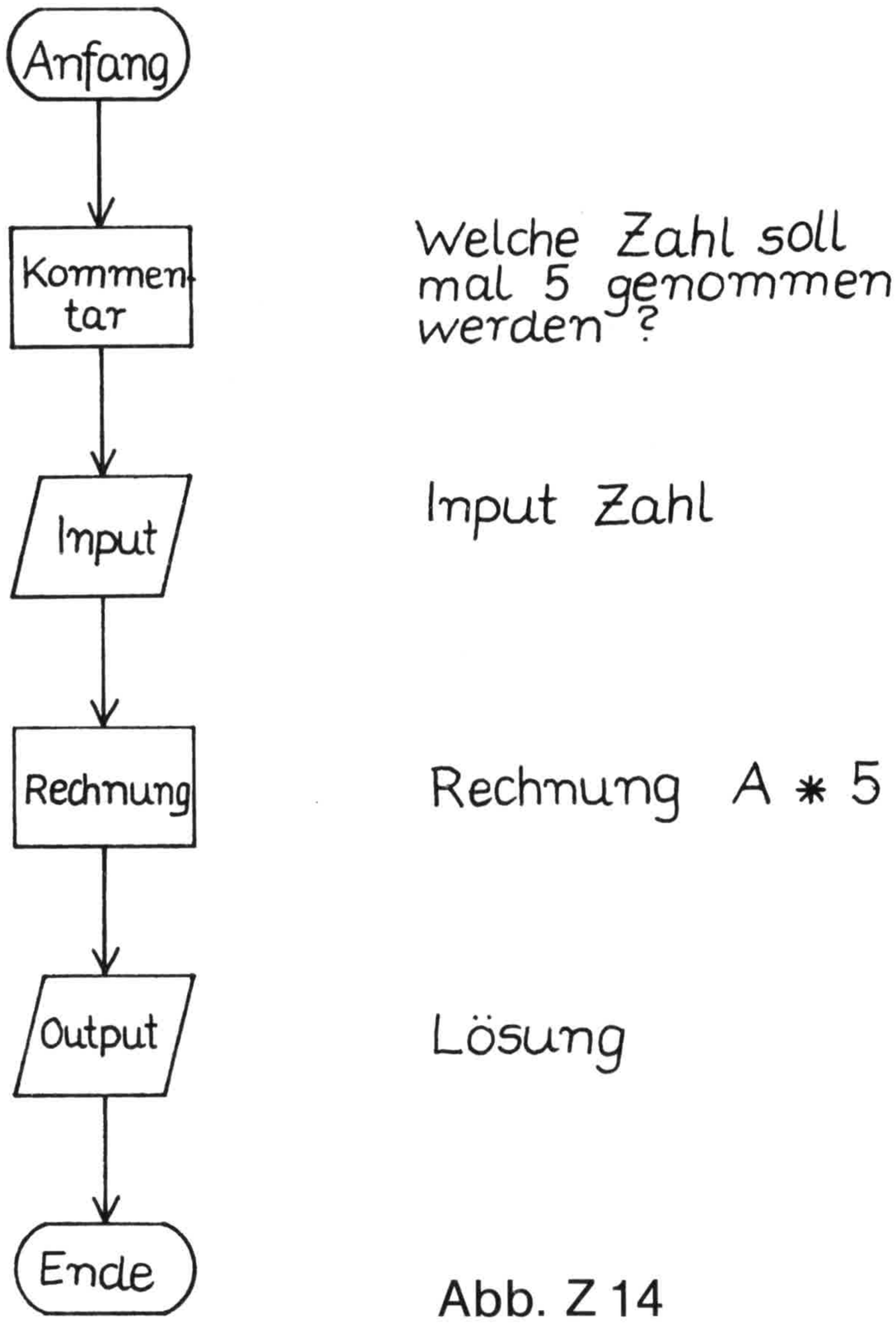


Abb. Z 14

Sicher ist dir aufgefallen, daß nicht jede BASIC-Zeile durch ein entsprechendes Symbol im Flußdiagramm wiedergegeben wird. Entscheidend ist, daß alle notwendigen Programmschritte im Ablaufplan auftauchen.

Umgekehrt kann zwar oft eine einzige Zeile genügen, um den Programmschritt vom Diagramm ins BASIC-Listing umzusetzen. Doch häufig benötigt man mehr. Vor allem der PRINT-Befehl ist da sehr heikel. Die auf dem Bildschirm angezeigten oder vom Drucker ausgegebenen Texte sollen ja gut lesbar sein.

Versuche nun, aus dem Flußdiagramm (aber ohne zu "spikken"!) wieder ein BASIC-Programm zu machen. Vielleicht sieht es ein bißchen anders aus als das ursprüngliche Programm. Aber Hauptsache ist, es läuft und liefert das gewünschte Ergebnis.

Möglicherweise war dir dieses Übungsbeispiel zu einfach. Dann nimm dir das Programm "Monatsnamen" vor, das wir im

letzten Kapitel entworfen haben. Probiere es, bevor du unseren Lösungsvorschlag in Abbildung Z 15 einfach übernimmst.

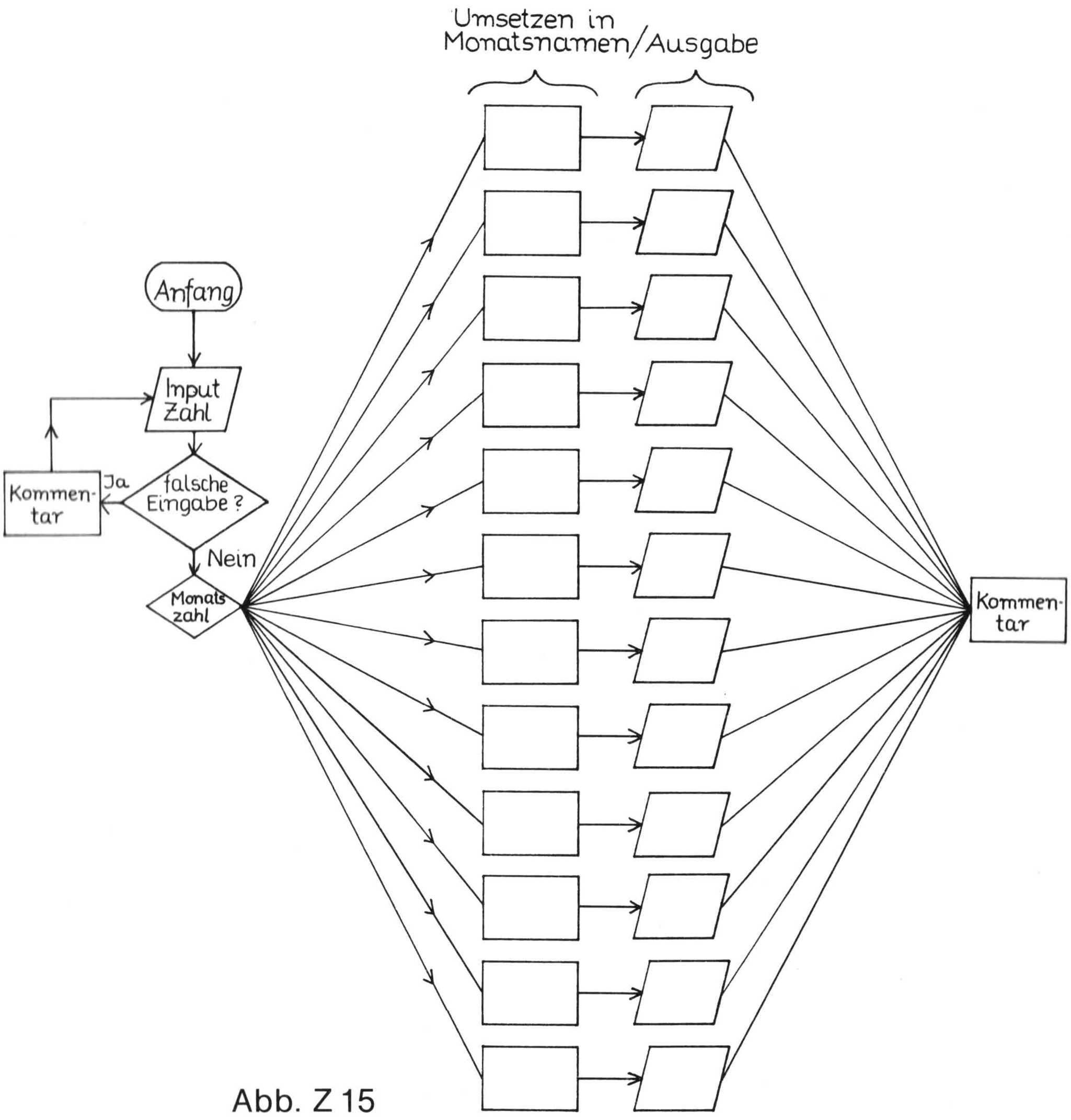


Abb. Z 15

Dem Profi über die Schulter geschaut

Zweifellos macht es gerade am Anfang enormen Spaß, sich einfach an den Computer zu setzen und ganz spielerisch ein bißchen herumzuwursteln. Man probiert dies und das, läßt das Listing immer wieder durchlaufen, um hier und da was einzufügen, umzutauschen oder wieder herauszunehmen. Oft entsteht dabei sogar ein funktionsfähiges, ja durchaus vorzeigbares Programm.

Allerdings murmeln erfahrene Programmierer beim Anblick solch improvisierter Schöpfungen nicht selten so etwas wie: "Na ja, ganz nett, aber ziemlicher 'Spaghetticode'." Bei näherem Hinschauen wird nämlich rasch deutlich, daß man vieles systematischer, übersichtlicher und einfacher hätte programmieren können. Und irgendwann hast du sicher vor, umfangreichere und leistungsfähigere Programme zu entwickeln. Dann merkst du bald, daß du mit dieser spontanen Arbeitsweise nicht mehr so einfach zum gewünschten Ergebnis kommst.

Daher empfiehlt es sich, so vorzugehen wie der berufsmäßige Programmierer. So ein Profi zerlegt die Arbeit am Programm in mehrere Schritte. Hierbei haben sich einige allgemeine Regeln bewährt. Sie lassen dennoch genügend Spielraum für den ganz persönlichen Stil und Geschmack.

Wir schlagen folgende Arbeitseinteilung vor:

I. Wir untersuchen das Problem

Überlege dir, was genau du mit deinem Programm erreichen willst. Zum Beispiel soll das Programm, das in Abbildung HC 4 auf Seite 16 aufgelistet ist, folgendes können: Es soll jede Zahl, die der Benutzer eingibt, mit 5 multiplizieren und das Ergebnis auf dem Bildschirm anzeigen.

Überlege dir bei diesem ersten Arbeitsschritt außerdem:

- Welche Rechenarten werden benötigt?
- Welche Programmierschritte sind erforderlich?
- Welche Eingaben müssen gemacht werden?

So wird dir gleich von vornherein klar, ob du für das von dir

angestrebte Programm bereits die erforderlichen Rechenfähigkeiten und Programmierkenntnisse besitzt. Du mußt außerdem abschätzen können, ob derjenige, der das Programm benützen soll (also der "Anwender"), überhaupt damit zurechtkommt. Denn in unserem Programmbeispiel muß der Anwender zumindest wissen, was eine Multiplikation ist.

Diese Anfangsüberlegungen ersparen es dir, daß du oder der Anwender irgendwann im Programm steckenbleiben – und das, obwohl du dir damit vielleicht schon so viel Mühe gegeben hast.

II. Wir beschreiben das Programm

Schildere jetzt genauer, was das Programm tun soll und vor allem *wie*.

Unser Programmbeispiel, das "Fünfer-Einmaleins", läßt sich folgendermaßen beschreiben:

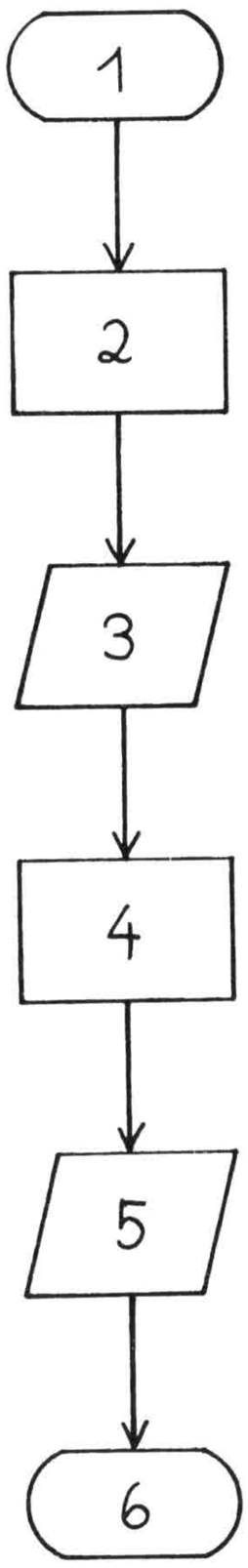
- Das Programm sagt dem Benutzer, daß es das Fünfer-Einmaleins beherrscht.
- Auf diese Information hin soll der Benutzer eine Zahl eingeben.
- Diese Zahl multipliziert der Computer mit fünf.
- Das Ergebnis dieser Berechnung gibt der Computer auf dem Bildschirm aus.

Dabei kannst du dir schon überlegen, welche BASIC-Befehle du verwenden willst. In diesem Beispiel wirst du die Ausdrücke *PRINT mit Anführungszeichen* und *PRINT ohne Anführungszeichen* sowie *INPUT* benötigen.

III. Wir entwerfen das Flußdiagramm

Der Ablaufplan veranschaulicht die Logik des Programms. Die Abbildung Z 14 von vorhin zeigt, welches Flußdiagramm sich aus unserem Programmbeispiel ergibt.

Der Übung halber empfiehlt es sich wieder, dieses Diagramm abzuzeichnen. Dabei solltest du die Symbole so groß machen, daß du die Beschreibung eines jeden Programmschrittes in den Sinnbildern eintragen kannst. Falls der Text so umfangreich wird, daß er nicht mehr in das Sinnbild paßt, kannst du ihn auch daneben schreiben und mit einer Nummer kennzeichnen. Diese Nummer trägst du dann auch in das Symbol ein. Das könnte etwa so aussehen, wie in Abbildung Z 16.



1 Anfang

2 Kommentar: Welche Zahl soll mal 5 genommen werden ?

3 Input: Eingabe der zu rechnenden Zahl

4 Rechnung: A * 5

5 Output: Ergebnis der Rechnung

6 Ende

Abb. Z 16

IV. Wir erstellen eine Variablenliste

Anhand des Flußdiagramms erkennst du, wie viele und welche Variablen im Programm erforderlich sind. Grundsätzlich gibt es zwei Möglichkeiten, Variablen zu benennen:

1. Wir halten uns ans ABC

a) Du gehst alphabetisch vor. Der Variablenname ist ein Buchstabe. Jede neue Variable erhält als Namen den nächsten Buchstaben im Alphabet. Hast du beispielsweise vier Variablen in deinem Programm vorgesehen, bekommen sie die Bezeichnung A, B, C, D.

b) Falls deine Variablennamen sich aus jeweils zwei Buchstaben zusammensetzen, sieht das bei sechs Variablen so aus: AA, AB, AC, AD, AE, AF;
oder so: BA, BB, BC, BD, BE, BF;
so ist es eher unpraktisch, AA, BA, CA, DA, EA, FA;
aber selbst so geht es: AA BB CC DD EE FF;
Dir fallen sicher noch etliche andere Möglichkeiten ein. Üblich sind jedoch die ersten beiden, also AA, AB, AC . . . und BA, BB, BC, . . .

c) Auch ein Buchstabe mit einer Zahl dahinter kann eine Variable darstellen, zum Beispiel:

A1, A2, A3, A4, A5, . . .
oder: B1, B2, B3, B4, B5, . . .
aber auch: A1, B1, C1, D1, E1, . . .
das ist schon fast ein wenig verwirrend:
A1, B2, C3, D4, E5, . . .
Wir empfehlen jedenfalls die ersten beiden Möglichkeiten, also A1, A2, A3, . . . oder B1, B2, B3, . . .

2. Die Variablennamen sprechen für sich selbst

Vielleicht findest du es interessanter oder praktischer, die Variablen so zu benennen, daß ihre Namen an deren Aufgabe im Programm erinnern. Doch muß man hierbei sehr aufpassen. Sonst kann dir nämlich passieren, daß du einen Variablennamen versehentlich zweimal vergibst. Allerdings kann man sich durch genaue Variablenlisten dagegen absichern. Dennoch erfordert diese Vorgehensweise viel mehr Aufmerksamkeit.

In unserem Beispiel könnte die (zugegebenermaßen äußerst kurze) Variablenliste folgendermaßen aussehen:
entweder so:

A = Vom Benutzer eingegebene Zahl, die vom Programm mit der Zahl fünf multipliziert wird
oder so:

BZ = Vom **B**enutzer eingegebene **Z**ahl, die vom Programm mit der Zahl fünf multipliziert wird.

In dem Programmbeispiel von Seite 18 käme in die Variablenliste ein zweiter Variablenname, nämlich:

K = Zähler für die Anzahl der Programmdurchläufe

oder

ZP = **Z**ähler für die Anzahl der **P**rogrammdurchläufe.

V. Wir übertragen den Ablaufplan in die Programmiersprache

Bei kleineren Programmen kannst du diesen Arbeitsschritt unmittelbar auf dem Bildschirm ausführen. Soll ein umfangreicheres Programm entstehen, entwickeln selbst die professionellen Programmierer das Programm keineswegs in einem Stück am Monitor (also auf dem Computerbildschirm). Statt dessen erarbeiten sie die einzelnen Programmbausteine auf dem Papier. Für diese Arbeit gibt es im Fachhandel sogar eigene Vordrucke. Die sehen ungefähr so aus wie in Abbildung "List 1" und erleichtern das Programm-Schreiben erheblich.

Zeilennummer	Anweisung
10	PRINT "ICH NEHME DIE ZAHL"
20	PRINT "NACH DEM FRAGEZEICHEN"
~~30~~	~~PRINT "MAL FÜNF~~
30	PRINT "MAL FUENF"
40	INPUT A
50	PRINT A*5
60	BEEP:RUN ZUM TOASTER AND RETURN NEXT
	~~INPUT KAK~~
70	OPEN KAKAU FOR NEXT IMBISS
80	GOTO 60
90	GAGA

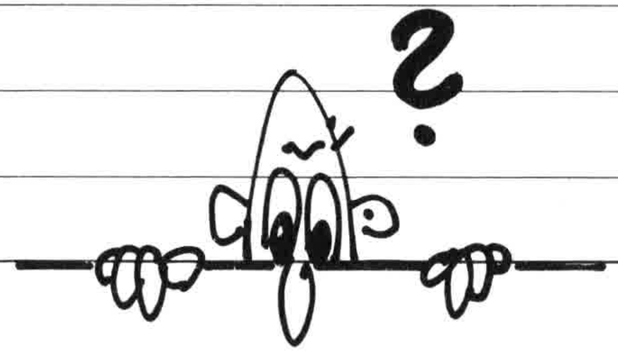


Abb. "List 1"

Den Lesern unserer ersten beiden Bände der Reihe "Computer kinderleicht" kommt dieses Formular sicher bekannt vor. In

beiden Büchern sind nämlich dem Anhang einige dieser Vordrucke beigefügt – so auch in diesem Band.

Einem fleißigen Programmierer werden die paar Seiten sicher kaum reichen. Natürlich könnten wir eine Menge Geld verdienen, wenn alle Programmierfreaks jedesmal das Buch neu kaufen würden, statt praktischerweise diese Vordrucke einfach zu fotokopieren. Aber uns leuchtet auch ein, daß sie ja nicht zigmal das gleiche Buch im Regal stehen haben wollen oder aus lauter Verlegenheit an den ganzen Freundeskreis verschenken möchten . . .

Sollte dir die Liste in dieser Form nicht zusagen, kannst du sie ohnehin nach deinen Wünschen selber herstellen.

Für diesen fünften Schritt auf dem Weg zum selbstentwickelten Programm verwenden die Fachleute einen speziellen Ausdruck. Das Übertragen der Programm-Ideen und des Flußdiagramms in eine Programmiersprache nennen die Computerprofis "Codieren".

VI. Wir geben das fertige Programm ein

In diesem Arbeitsschritt lauern noch etliche Fehlerquellen. Daher empfehlen wir dringend, das Programm in mehrere übersichtlich gegliederte Blöcke aufzuteilen. Diese Blöcke kannst du nun gesondert eingeben und zweckmäßigerweise gleich durchtesten. Nützlich dürften auch unsere "Tips & Tricks für das Eingeben der Listings" sein. Sie sind in diesem Band wieder dem Anhang beigefügt.

Beim Abschreiben vom Papier solltest du dir eine kleine visuelle Hilfestellung gönnen, nämlich ein "Zeilen"-Lineal. Diesen bewährten Trick mag niemand mehr missen, der sich schon etliche Male darüber ärgern mußte, daß er dieselbe Programmzeile versehentlich zweimal eingetippt oder auch übersehen hat. Ohne Zeilenlineal geschieht es auch leicht, daß du zwar die richtige Programmzeile eingibst, aber die falsche Zeilennummer davorsetzt (oder umgekehrt).

Am besten sind die durchsichtigen Lineale. Es gibt welche (natürlich die etwas teureren), die besitzen sogar besondere Skalen für die gängigen Zeichenabstände auf ausgedruckten Listings. Damit entfällt das lästige Zeichen-Zählen beim Abtippen von sogenannten "hardcopies" oder "printouts", also Programmausdrucken.

Zur Not tut es ein Geo-Dreieck genausogut. Mancher Könner rümpft darüber allerdings die Nase und kauft sich statt dessen lieber Disketten. Derlei "edelvoller Schnickschnack" sei doch bloß was für Leute, die mit Profi-Zubehör Eindruck schinden wollen. Nun ja – jeder wie er mag.

Auf jeden Fall sollten wir unseren geplagten Augen größtmögliche Erleichterung verschaffen, sei es durch derartige oder auch anderweitige Hilfsmittel. Dazu gehören eigentlich auch blendfreie Beleuchtung, optimales Schreibgerät und körpergerechte Sitzhöhe – bis hin zu Klemmbrettern an Schwenkarmen zum Festheften der Papiervorlagen. (Sehr komfortabel! Bewahrt einen vor Nackenschmerzen und ist ziemlich einfach aus alten Gelenkleuchten zu basteln!)

VII. Austesten und "debugging" – Wir suchen nach Fehlern und geben dem Programm den letzten Schliff

In dieser Arbeitsstufe kannst du immer noch viel verändern und verbessern. Falls das fertige Programm jetzt zwar fehlerfrei läuft, dir aber noch nicht richtig gefällt, dann GOTO Arbeitsschritt 1, 2 oder 3 AND überlege, was du noch verbessern oder ergänzen willst. (AND – gesprochen "änd" – ist das englische Wort für "und".)

Irgendwann kannst du dich zufrieden zurücklehnen und behaupten: "Mein Programm steht jetzt." Nach den letzten Probeläufen hast du die endgültige Version auf Diskette abgespeichert. Du hast auf einer zweiten Diskette oder Kassette eine Sicherheitskopie angelegt, eine sogenannte "backup" (sprich "bäckap").

Trotzdem ist die Arbeit noch nicht ganz zu Ende. Du weißt ja, wie empfindlich Disketten sind.

Stell dir vor, du verschüttest versehentlich Limo, und die Disketten sind verklebt. Du könntest auch vergessen haben, sie in die Schutzhülle zurückzustecken, so daß Radierprökel oder Schmutzteilchen auf die Oberfläche geraten. Ein ungewollter Fingerabdruck auf der magnetischen Oberfläche – Datenverlust! Ein Telefon darauf gestellt, jemand ruft an, und das magnetische Streufeld der Klingel erzeugt Datenschrott auf den Sektoren.

Du mußt ja gar nicht erst versucht haben, das bereits aufgeklebte Etikett mit Kugelschreiber zu beschriften. Es reicht,

die Diskette versehentlich auf der winterlich voll aufgedrehten Zentralheizung zu parken oder auf den Lüftungsschlitzen des Diskettenlaufwerks abzulegen.

Ganz zu schweigen von den filmreifen Katastrophen: wenn etwa dein kleiner Bruder beide Disketten, nämlich das Original (die sogenannte "Master"-Diskette) und die "backup" zum Frisbee-Spielen verwendet. Oder euer junger Hund hat darauf herumgekaut. Oder deine Mutter hat sie als Untersetzer für den Topf mit Kartoffelbrei verwendet (oder der Großvater als Bierdeckel).

Kann sein, daß du von all dem verschont bleibst. Dann können immer noch plötzlich defekte Floppylaufwerke, Stromausfälle und ähnliche Schicksalsprüfungen die Freude am Erfolg vermiesen.

Das alles (vielleicht auch nur einiges davon) könnte schlimmstenfalls eintreten, bevor dir der Drucker von dem Programm eine Hardcopy ausgegeben hat! Wohl dem, der vorher – notfalls handschriftlich, aber lesbar – die lauffähige Fassung seines Programms auf einem Block notiert hat. Mit zunehmender Erfahrung wirst du darauf ohnehin nicht mehr verzichten wollen: solide *Dokumentationen!*

VIII. Wir legen eine umfassende Programm-Dokumentation an

Der vorbildliche Programmierer fertigt sich nicht nur übersichtlich abgefaßte, schriftliche Unterlagen zum Aufbau seines Programms an. Er notiert auch in möglichst verständlicher Form, was für den Programmbenutzer zur Handhabung des Programms wichtig ist.

Dazu gehört wenigstens:

a) eine Beschreibung des Programmablaufs
b) ein Vorschlag für den Test deines Programms
c) eine Liste möglicher Fehler.

Sind die Programme komplizierter aufgebaut, mußt du noch weitere Erklärungen hinzufügen. Nehmen wir an, in deinem Programm werden "Dateien" benützt. Diese sollten ausführlich und lückenlos beschrieben sein.

Nachdem du nun all diese Punkte beherzigt hast, kann eigentlich nichts mehr schiefgehen. Du hast dein Programm erfolgreich ausgetestet. Du hast ausgefallene und ungewöhn-

liche Eingaben ausprobiert. Die Dokumentation ist erstellt und liegt säuberlich abgeheftet im entsprechenden Ordner. Jetzt kannst du guten Gewissens jubeln:

”DAS PROGRAMM IST FERTIG!“

Ein Musterbeispiel: Zoll in Zentimeter umrechnen

Entwickeln wir nun ein weiteres Programm. Dabei kommen sämtliche Arbeitsschritte vor, die wir bislang besprochen haben.

Du beginnst mit der Problembeschreibung:

Gerade jemand, der sich mit der Computerei abgibt, begegnet irgendwann dem Längenmaß "Zoll". Spätestens, wenn du Disketten als Datenträger verwendest, kommst du an dieser Maßeinheit nicht mehr vorbei. Eigentlich gilt dieses Längenmaß schon als längst veraltet. In der Technik wird es trotzdem immer noch häufig verwendet. Am weitesten ist es noch in den englischsprachigen Ländern verbreitet, also Großbritannien, USA, Kanada, Australien usw. Aber auch hierzulande triffst du es an.

Als Commodore-Anwender arbeiten wir mit $5^1/_4$-Zoll-Disketten. Vielleicht hast du auch schon von anderen Diskettengrößen gehört. Es gibt, als älteste Norm, 8-Zoll-Disketten und kleinere Größen, wie $3^1/_2$ Zoll. Die großen Hersteller versuchen zur Zeit, die Disketten-Durchmesser weiter zu verringern – bei immer größerem Speichervermögen! Dadurch will man vor allem bei tragbaren Computern handlichere Geräteabmessungen erreichen.

Nicht nur in der Computerei gebraucht man Zoll. Gewindegrößen, die Durchmesser von Reifen, von Wasser- und Gasrohren, die Breite von Videobändern und vieles mehr werden in Zoll gemessen.

Offenbar bietet ein Programm, das Zoll in uns wohlvertraute Zentimeter umrechnet, einigen praktischen Nutzen.

Ein Zoll entspricht ungefähr 2,6 Zentimeter. Das sind etwa 26 Millimeter. Andersherum: Ein Zentimeter entspricht etwa 0,38 Zoll. Miß das selbst an einer Diskette nach. Um jedes Risiko auszuschließen, nimm möglichst eine bereits ausrangierte oder fertige dir von der Diskette eine Fotokopie an.

Sicher kommst du ebenfalls zu unserem Ergebnis: Eine $5^1/_4$-Zoll-Floppy ist zwischen 13,6 bis 13,7 cm hoch und breit. Wenn

du zur Probe nachrechnest: $5^1/_4$ mal 2,6 cm ergeben 13,65 cm. Die gemessenen Zentimeter kannst du nun wieder in Zoll umrechnen. Das läßt sich zur Abwechslung schriftlich machen oder bequemer im Direktmodus auf deinem "Rechner", dem Computer. Was dann auf dem Bildschirm erscheint, siehst du in Abbildung HC 36.

```
?13.6 * 0.38
 5.168
```

Abb. HC 36

Du erhältst ein "angenähertes" Ergebnis, weil auch die Zahl 0,38 kein ganz genauer Wert ist, sondern gerundet. Etwas genauer gemessen beträgt nämlich ein Zentimeter (nicht ganz) 0,384615386 Zoll. Je mehr Stellen wir beim Umrechnen hinter dem Dezimalkomma haben, desto genauer wird das Ergebnis. Für unsere Zwecke hingegen dürfte der Näherungswert von 0,38 hinreichen.

Wir wollen ein möglichst benutzerfreundliches Programm schreiben. Es soll dem Anwender sowohl Zentimeter in Zoll als auch Zoll in Zentimeter umrechnen. In beiden Fällen muß der Computer multiplizieren.

Die Programmbeschreibung

Wagen wir uns also an die Programmbeschreibung!

Zuallererst wollen wir dem Programm einen Namen geben. Ob du es "ZOLL IN ZENTIMETER UMRECHNEN" nennst oder nur "ZOLL IN ZENTIMETER", bleibt dir überlassen. Den gewählten Namen trägst du jedenfalls mit REM in die erste Programmzeile ein. Unter diesem Namen speicherst du das Programm auf den Datenträger, sei es Diskette oder Kassette.

Im nächsten Schritt soll der Computer den Bildschirm löschen, damit der Programmbenutzer die Übersicht nicht verliert.

Dann soll die "Maschine" den Anwender freundlich begrüßen, indem sie ihm einen

"GUTEN TAG"

wünscht.

Nichts spricht dagegen, daß der Computer den Benutzer ausführlicher grüßt:

"EINEN WUNDERSCHOENEN GUTEN TAG!
ICH BEGRUESSE SIE ZUM UMRECHNUNGSPROGRAMM
ZOLL/ZENTIMETER!"

Doch dann kommt der Computer sofort zur Sache. Er kann ja nicht nur Zoll in Zentimeter umrechnen, sondern auch Zentimeter in Zoll. Genauer gesagt: Das Programm soll aus zwei Teilen bestehen. Der eine Programmzweig rechnet Zoll in Zentimeter um, der andere Zweig rechnet Zentimeter in Zoll um. Folglich muß der Computer erfahren, welche dieser beiden Möglichkeiten der Anwender beanspruchen möchte. Also ist eine Bildschirmmeldung fällig. Die erste Frage, die das Programm an den Benutzer stellt, lautet:

"WOLLEN SIE ZOLL IN ZENTIMETER UMRECHNEN?
DANN TIPPEN SIE BITTE DIE ZIFFER 1."

Wir legen das Programm nämlich so an, daß der Benutzer seine Antwort als Ziffer eingeben muß. Das vereinfacht den Programmieraufwand.

Die beiden nächsten Programmschritte bieten umgekehrt die Umrechnung von Zentimeter in Zoll an. Wünscht der Anwender, daß der Computer in dieser Richtung rechnet, soll er die Ziffer 2 eintippen. Entsprechend formulieren wir die Bildschirmmeldung.

Hat der Benutzer sich entschieden, teilt er dies dem Computer in der folgenden INPUT-Zeile mit. Hierzu tippt er entweder die 1 oder die 2 ein. Nach dieser Eingabe soll das Programm in den betreffenden Programmteil abzweigen.

Wichtig ist nun, daß der Programmablauf nicht durch unsinnige Eingaben des Anwenders gestört werden kann. Wir müssen somit vor diese "bedingte Verzweigung" einen Schutz gegen unerwünschte Eingaben legen. Unerwünschte Eingaben sind alle Zeichen oder Ziffern, die anders als 1 und 2 lauten. Gibt der Benutzer zum Beispiel die Ziffer 3 ein oder tippt "NULL BOCK HEUTE", soll er antworten:

"DIESE ANTWORT VERSTEHE ICH NICHT."

Jetzt erst kommen die eigentlichen Rechenvorgänge. Zuerst soll der Benutzer die Anzahl der umzurechnenden Zoll (beziehungsweise Zentimeter) eintippen. Wir gehen davon aus, daß er den Computer nicht ärgern will und die Länge nicht in Worten,

sondern schön brav in Zahlen eingibt. Daher verzichten wir an dieser Stelle auf Schutzvorkehrungen gegen unsinnige Eingaben.

Als Grenze für die größtmögliche umzurechnende Länge ist grundsätzlich nur der verfügbare Speicherplatz im Computer selbst ausschlaggebend. Aber nehmen wir an, daß unser Anwender ganz normale Längenangaben umrechnen lassen will und nicht die Entfernung von der Erde zur Sonne.

Die Umrechnung selbst bringen wir gleich als OUTPUT auf den Bildschirm. Der Computer muß die eingegebene Zahl entweder mal 2,6 nehmen (Zoll in Zentimeter) oder mit 0,38 multiplizieren.

Dabei mußt du berücksichtigen, daß auch der Computer die Dezimalstellen nicht durch das sonst übliche Komma trennt, sondern durch einen Punkt. Vielleicht ist dir das bereits bei Taschenrechnern aufgefallen. Die Zahl 2,6 zeigt der Computer als 2.6 an, und die Zahl 0,38 wird als 0.38 angezeigt. Demgemäß tippst du 2,6 als 2.6 ein.

Schließlich soll das Programm noch eine "Endabfrage" enthalten. Der Benutzer entscheidet, ob er das Programm noch einmal anwenden will oder es beenden möchte. Diese Möglichkeit bieten wir dem Anwender anhand einer weiteren INPUT-Zeile. Er kann seine Entscheidung wieder in Form der Ziffern 0 und 1 dem Computer mitteilen.

Wir könnten hier aber auch anders vorgehen. Gibt der Anwender die Ziffer 1 ein, steht das Programm für den nächsten Durchgang bereit. Jede andere Eingabe hingegen soll bewirken, daß sich das Programm mit einer passenden Bildschirmmeldung verabschiedet. So schützen wir es bei diesem INPUT zugleich gegen unwillkommene Eingaben.

Damit ist unsere Programmbeschreibung vollständig. Für dich als Anfänger wollten wir jede wichtige Einzelheit berücksichtigen. Daher ist sie sehr ausführlich und entsprechend umfangreich geraten. Mit etwas mehr Übung kannst du so eine Beschreibung auch viel knapper gestalten, ohne etwas Wesentliches auszulassen.

Das Flußdiagramm

Als nächstes erstellen wir das Flußdiagramm. Hierbei halten wir uns an die eben erarbeitete Programmbeschreibung.

Nach dem Start "arbeitet" das Programm, indem es den Benutzer über sich selbst informiert (Symbol "START"). Es teilt ihm mit, wie es heißt und welchen Zweck es erfüllt (VERARBEITUNG).

Danach erwartet der Computer die erste Eingabe. Er muß wissen, ob der Anwender Zoll in Zentimeter oder Zentimeter in Zoll rechnen möchte (INPUT). Dabei müssen wir das Programm gegen falsche Eingaben schützen (VERARBEITUNG, ENTSCHEIDUNG). Die Eingabe entscheidet, welchen der beiden möglichen Wege das Programm weiterverfolgen soll (ENTSCHEIDUNG).

Bis jetzt sieht das Flußdiagramm so aus wie in Abbildung Z 17.

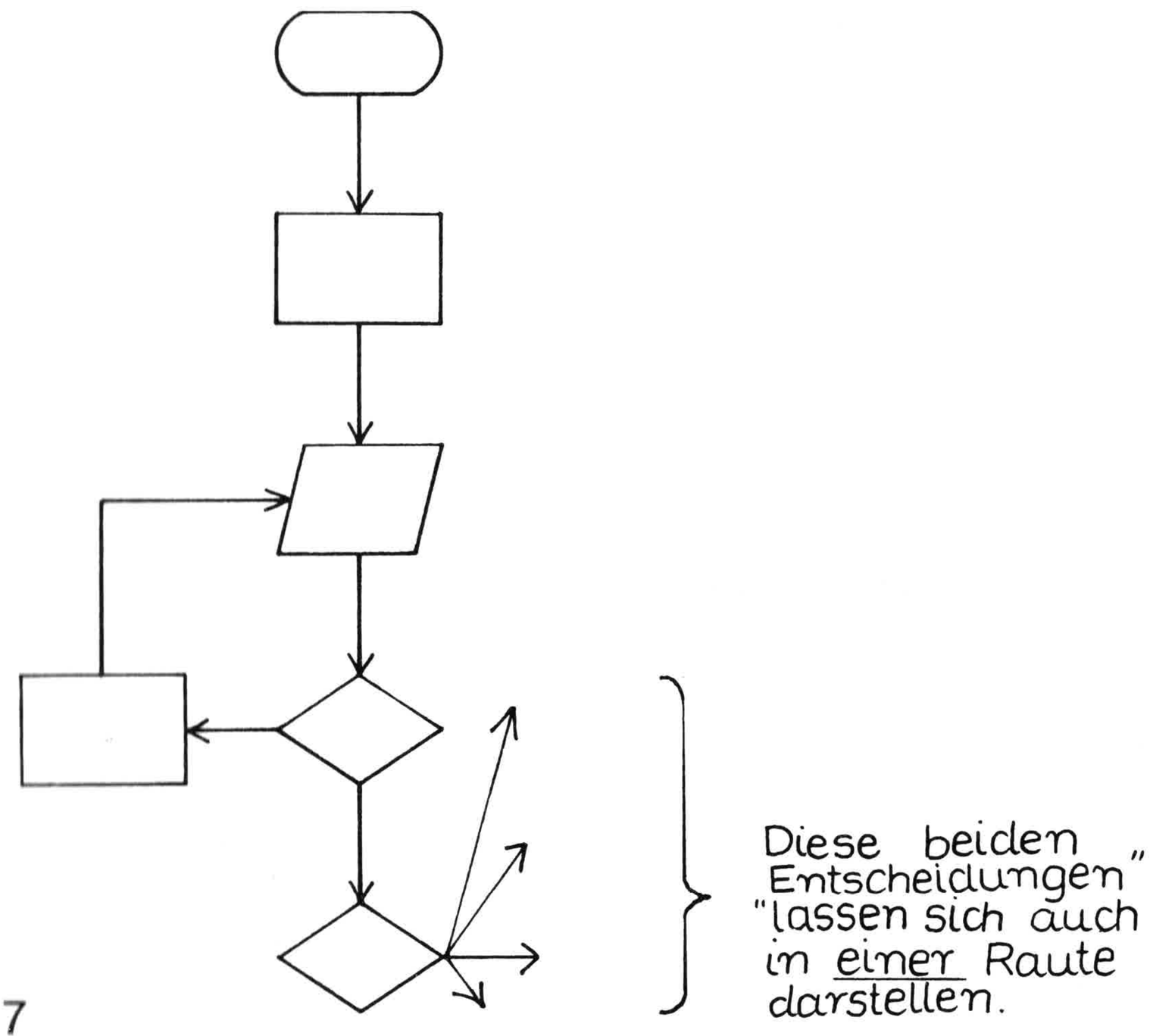


Abb. Z 17

Jeder der beiden Programm-Äste verlangt nun zunächst eine erneute Eingabe (zwei INPUT-Symbole). Diesmal erwartet der Computer jene Zahl, die umgerechnet werden soll.

Sodann folgt auf jedem Ast die entsprechende Rechenoperation (VERARBEITUNG) und anschließend die jeweilige Ausgabe. (Für OUTPUT verwenden wir gleichfalls das *Parallelogramm.)* In Abbildung Z 18 siehst du, wie der Programmablaufplan mittlerweile aussehen könnte.

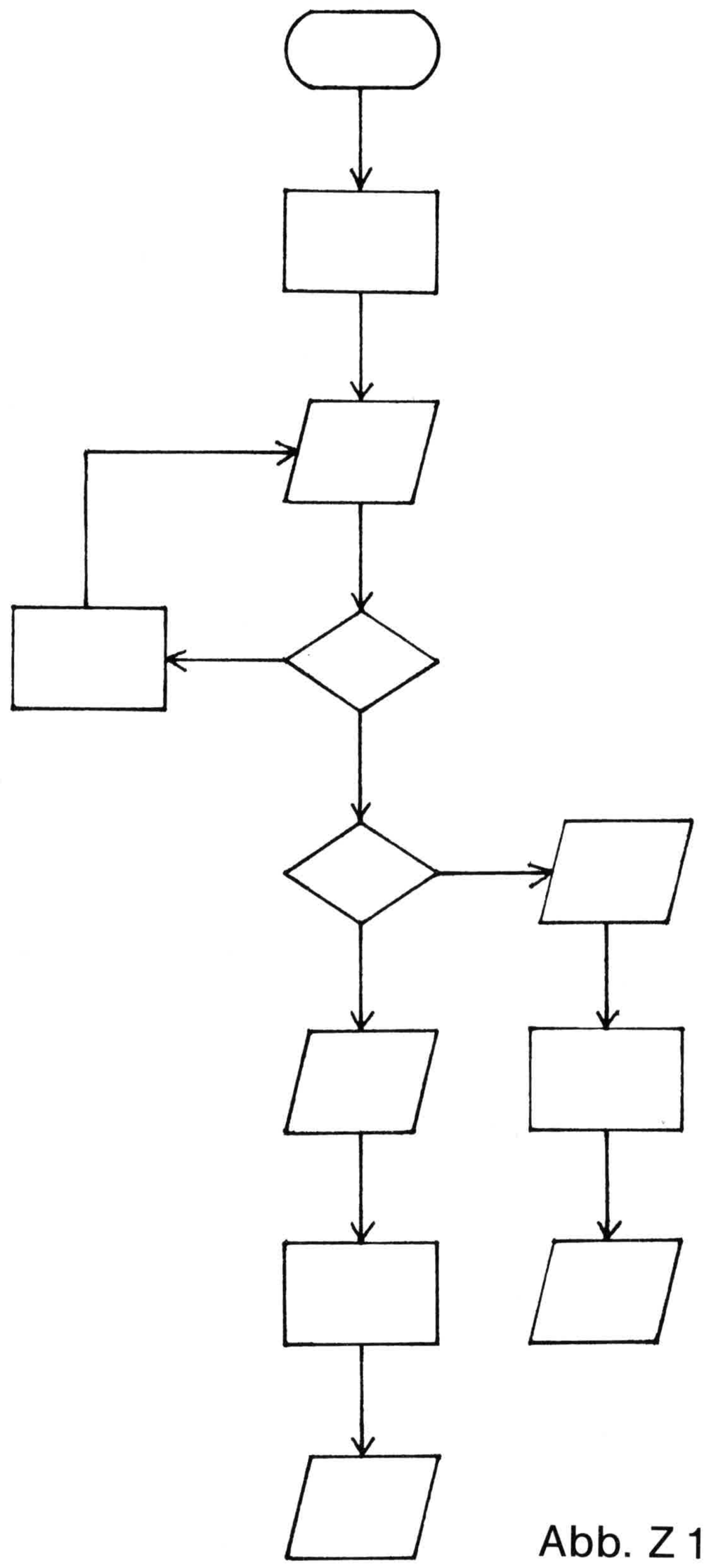


Abb. Z 18

Bei der Endabfrage werden die beiden Äste des Programms wieder zusammengeführt. Will der Benutzer noch eine weitere Berechnung durchführen lassen? (EINGABE).

Antwortet er (ENTSCHEIDUNG) mit der Ziffer 1, so bedeutet das "JA". Das Programm kehrt daraufhin an jene Stelle zurück, an der vom Benutzer die Eingabe erfolgen muß, in welche Richtung umgerechnet werden soll.

Wenn er nicht mit der Ziffer 1 antwortet, also mit 0 oder irgendeiner Unsinnseingabe, so bedeutet dies "NEIN". Dann soll das Programm mit einem Abschiedsgruß enden (ENDE).

Das fertige Flußdiagramm könnte nun aussehen wie in Abbildung Z 19.

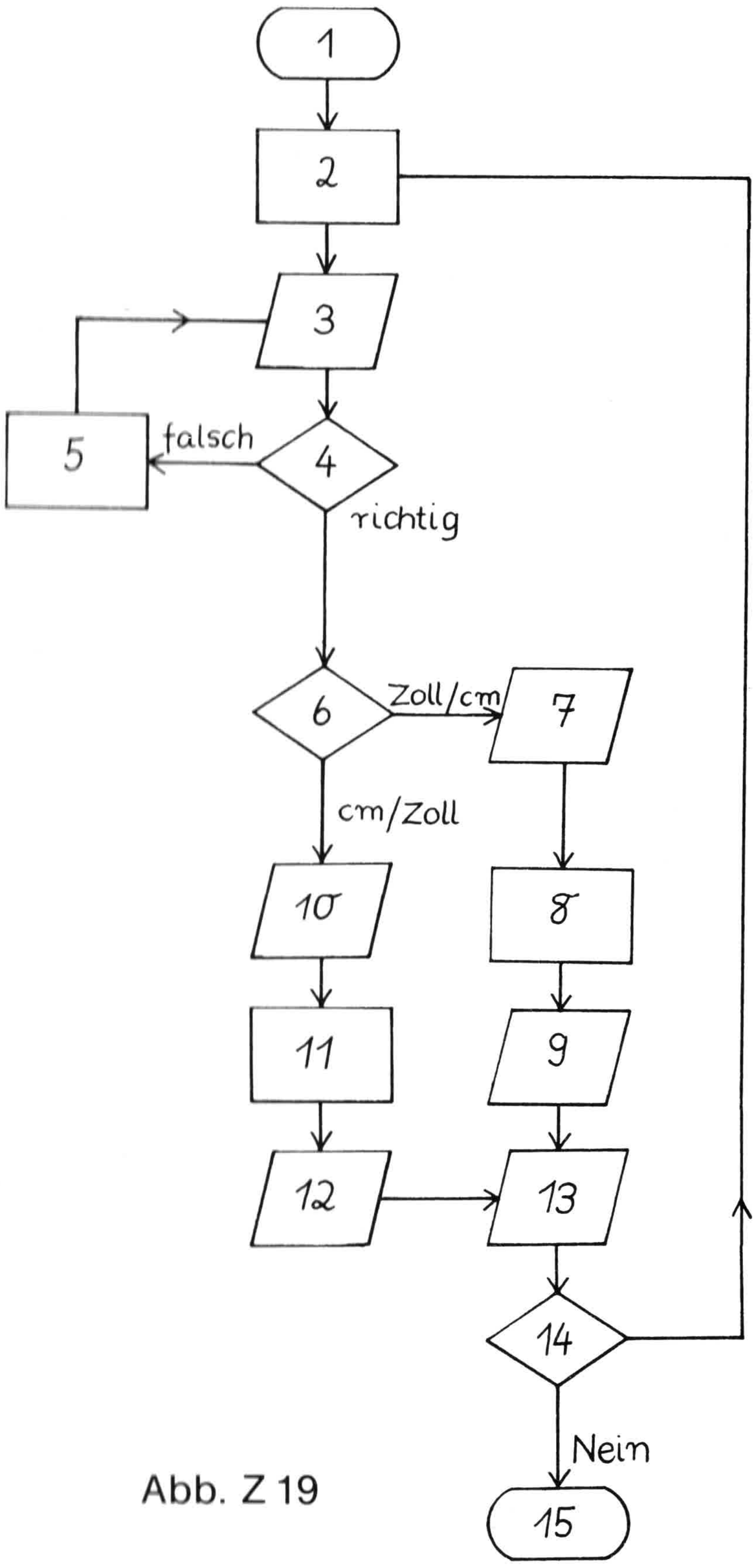


Abb. Z 19

1 Start

2 Kommentar

3 Eingabe

4 Eingabe falsch / richtig

5 Kommentar

6 cm / Zoll oder Zoll / cm

7 Eingabe: Zahl → Zoll / cm

8 Rechnung Zoll / cm

9 Ausgabe: Zoll / cm

10 Eingabe: Zahl → cm / Zol

11 Rechnung cm / Zoll

12 Ausgabe cm / Zoll

13 Eingabe noch eine Rechnung ?

14 Eingabe Ja / Nein ?

15 Ende

Die Variablenliste

Zum Programmablaufplan legen wir noch eine Variablenliste an. Sie umfaßt insgesamt drei Variablen mit den Variablennamen A, B und C. Selbstverständlich könntest du sie gemäß unseren Vorschlägen von vorhin auch anders bezeichnen.

A = 1 entspricht Zoll in Zentimeter
A = 2 entspricht Zentimeter in Zoll
B = umzurechnender Wert (nur Zahl als Eingabe!)
C = 1 Wiederholung des Programms
C ≠ 1 Ende

Das Codieren

Das Flußdiagramm in BASIC umzusetzen bereitet dir sicher keine nennenswerten Schwierigkeiten. Ausgesprochen lehrreich ist es, wenn du neben die Sinnbilder des Programmablaufplans gleich die entsprechenden BASIC-Zeilen notierst. Hoffentlich hast du das Diagramm hierfür auf ein genügend großes Blatt gezeichnet. Abbildung Z 20 zeigt unsere Fassung.

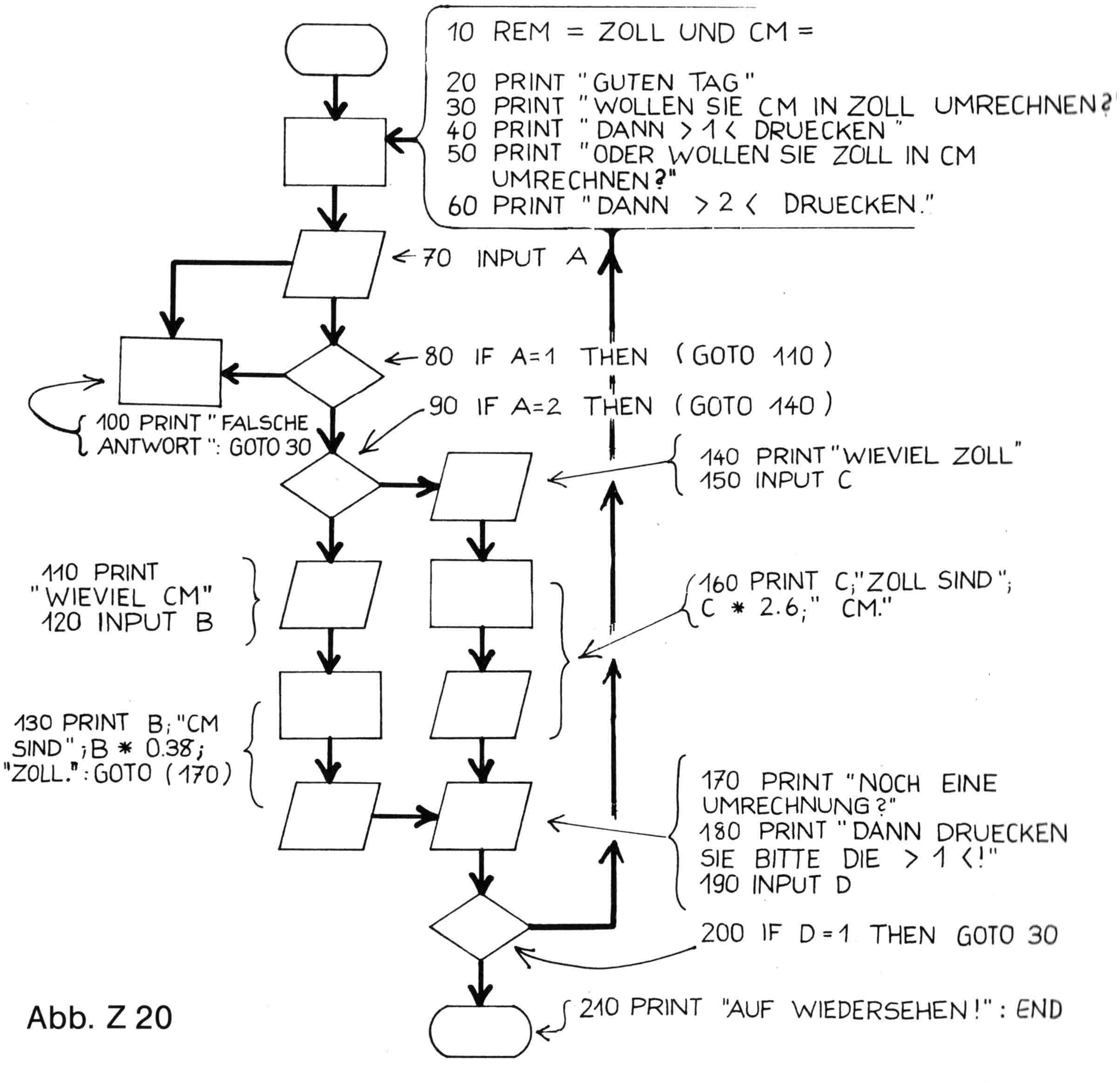


Abb. Z 20

Die neben deinem Flußdiagramm eingetragenen BASIC-Programmzeilen kannst du nun auf das Listing-Papier übertragen. Abbildung HC 37 zeigt unser Listing. Du siehst, unsere Version ist nicht übermäßig aufwendig. Wenn du mehr "Lametta" dran-

hängen willst, steht es dir frei, die PRINT-Zeilen nach deinen Vorstellungen auszugestalten.

```
10 REM * ZOLL UND CM *
20 PRINT "␣GUTEN TAG."
30 PRINT:PRINT "WOLLEN SIE CM IM ZOLL UMRECHNEN?"
40 PRINT "DANN DRUECKEN SIE DIE >1<."
50 PRINT:PRINT "ODER WOLLEN SIE ZOLL IN CM UMRECHNEN?"
60 PRINT "DANN DRUECKEN SIE DIE >2<."
70 INPUT A
80 IF A=1 THEN GOTO 110
90 IF A=2 THEN GOTO 140
100 PRINT "DIESE ANTWORT VERSTEHE ICH NICHT!":GOTO 30
110 PRINT "WIEVIEL CM"
120 INPUT B
130 PRINT B;" CM SIND ";B*0.38;" ZOLL.":GOTO 170
140 PRINT "WIEVIEL ZOLL"
150 INPUT C
160 PRINT C;" ZOLL SIND ";C*2.6;" CM."
170 PRINT "NOCH EINE UMRECHNUNG?"
180 PRINT "DANN DRUECKEN SIE BITTE DIE >1<."
190 INPUT D
200 IF D=1 THEN GOTO 30
210 PRINT "AUF WIEDERSEHEN!":END
```

Abb. HC 37

Das Eintippen

Gib jetzt das Programm über die Tastatur in den Computer ein und teste es. Wenn du bei den nun folgenden Probeläufen jeden "bug" (Fehler) herausgefischt hast, kannst du nach dem "debugging" jemand anderen das Programm durchprobieren lassen. Auf diese Weise erkennst du, ob zum Beispiel die Bildschirmmeldungen verständlich formuliert sind.

Die Dokumentation

Für das fertige Programm brauchst du noch eine kleine Dokumentation. Hier dürften wohl eine knappe Programmbeschreibung und der bereits erstellte Programmablaufplan genügen. Damit hast du die Programmierarbeit erfolgreich abgeschlossen.

Zum Abschluß dieses Kapitels noch eine knifflige Aufgabe:

In Abbildung HC 38 siehst du das Listing zum "MAX' EINMALEINS"-Programm. Dieses Programm befindet sich auch auf der Diskette zum Buch.

```
10 REM *** EINMALEINS ***
20 PRINT "␣GUTEN TAG!"
30 PRINT "␣ICH BIN DAS EINMALEINS-UEBUNGSPROGRAMM"
40 PRINT "VON ____________________"
50 PRINT:PRINT:PRINT
60 PRINT "DARF ICH FRAGEN, WIE ALT DU BIST"
70 INPUT A
80 IF A<4 THEN PRINT "DAS PROGRAMM IST ZU SCHWER FUER DICH":END
90 IF A>120 THEN PRINT "MEINE TASTATUR IST ZU KLEIN FUER ELEFANTEN!":END
100 IF A<19 THEN GOTO 170
110 PRINT "␣SO, SIE WOLLEN MEIN PROGRAMM BENUTZEN?"
120 PRINT "DAS KOSTET SIE 50 PFENNIG!"
130 PRINT "TIPPEN SIE EINE 1 EIN,"
140 PRINT "WENN SIE ZAHLEN WOLLEN."
150 INPUT B
160 IFB<>1 THEN PRINT "DANN EBEN NICHT. TSCHUESS.":END
170 PRINT "␣JETZT FANGEN WIR AN!"
180 PRINT "WELCHES EINMALEINS WOLLEN WIR UEBEN"
190 INPUT E
200 IF A<9 AND E>10 THEN PRINT "DAS IST WOHL ZU SCHWER":GOTO 180
210 IF E<21 THEN GOTO 270
220 PRINT "WIRKLICH?"
230 PRINT "MACH ERST MAL EINE TEST-AUFGABE"
240 PRINT "WIEVIEL IST 22 * 17"
250 INPUT T
260 IF T<>374 THEN PRINT "LEIDER FALSCH!":GOTO 180
270 C=1
280 PRINT:PRINT:PRINT
290 PRINT "AUFGABE:"
```

Hier kann dein Name Stehen!

```
300 PRINT C;"*";E;"="
310 INPUT F
320 IF F<>C*E THEN PRINT "LEIDER FALSCH. NOCHMAL!":GOTO280
330 PRINT "RICHTIG"
340 C=C+1:IF C<21 THEN 280
350 PRINT "{UUUUU}DAS GROSSE EINMALEINS MIT";E;"GESCHAFFT"
360 PRINT:PRINT:PRINT "NEUES SPIEL? [1]=JA,[2]=NEIN"
370 INPUT Y
380 IF Y=1 THEN GOTO 170
390 IF Y=2 THEN PRINT "TSCHUESS!":END
400 PRINT "WAS DENN NUN? [1]=JA ODER [2]=NEIN":GOTO 370

READY.
```

Abb. HC 38

Den Ablaufplan zum Programm findest du in Abbildung Z 21.

In diesem Diagramm haben wir sämtliche Hinweise und Texte weggelassen. Deine Aufgabe ist es, sie richtig einzufügen.* Kannst du auch anhand des Programmablaufplans und mit Hilfe des Listings eine Programmbeschreibung erstellen und schließlich eine Dokumentation?

Und falls du das Programm noch frisieren willst – nur zu! Es bleibt alles deinem persönlichen Programmstil überlassen. Doch sollten dann sämtliche Änderungen zugleich in die Programmbeschreibung und Dokumentation übernommen werden.

Wir sind gespannt darauf, welche Programm-Ideen dir dazu einfallen. Schreib uns, wenn du magst, an die vorhin angegebene Adresse!

---

* Die Lösung findest du auf Seite 118.

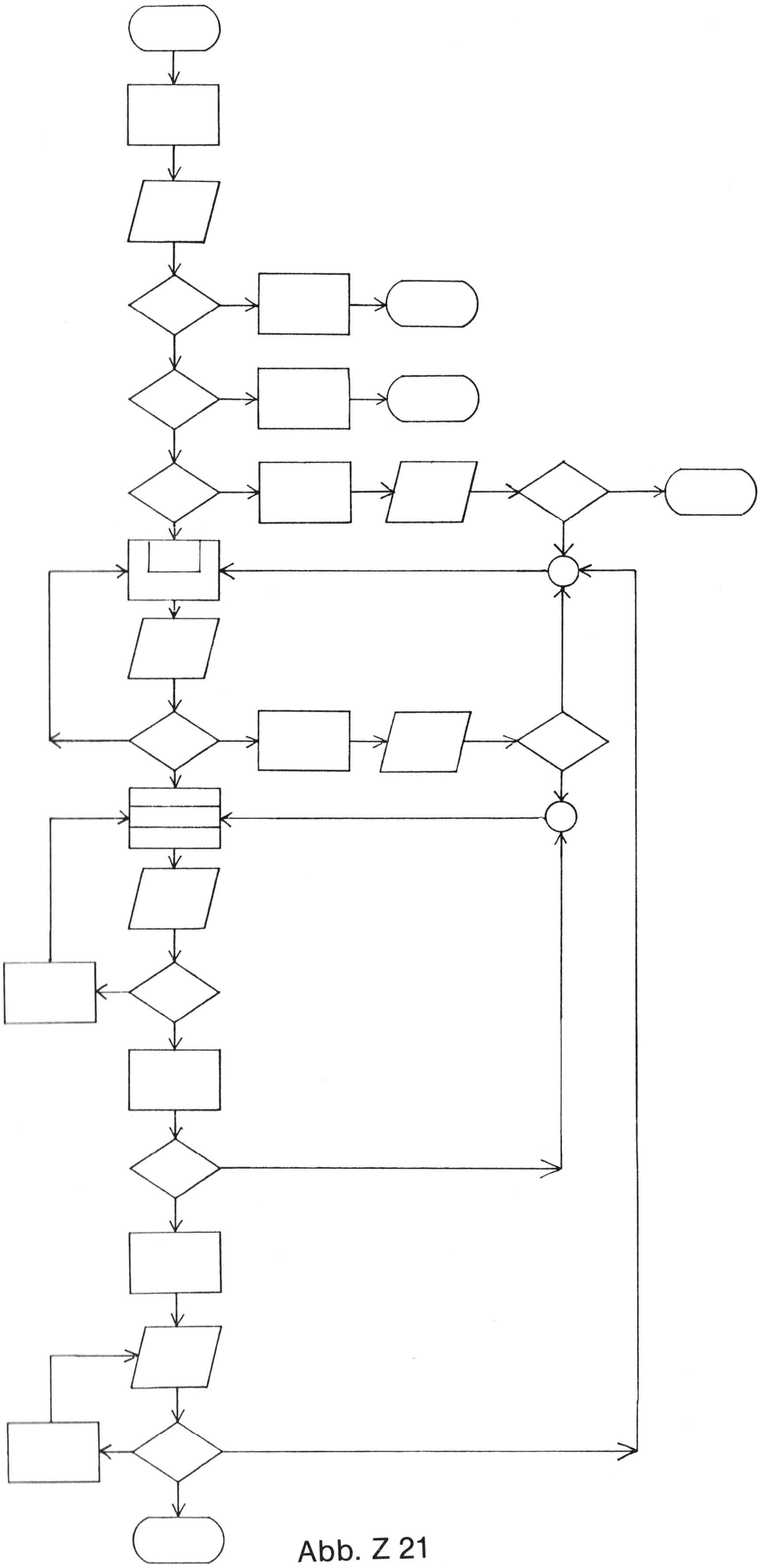


Abb. Z 21

Die elektronische Liebeserklärung

Sobald der Computer *Zahlen* verarbeiten sollte, haben wir innerhalb eines Programms Variablen verwendet. Eine Variable, die an Stelle einzelner Zahlen ("Nummern") steht, nennen wir auch *numerische* Variable.

Der Computer kann jedoch nicht nur Zahlen verarbeiten, sondern auch Buchstaben, ja sogar Wörter und ganze Sätze. Auch Buchstaben, Wörtern und Sätzen können wir nämlich Stellvertreter zuordnen. Die Stellvertreter für Zahlen, also die numerischen Variablen, haben wir meist aus einem oder zwei Buchstaben gebildet.

Stellvertreter für Buchstaben, Wörter oder ganze Sätze sehen etwas anders aus. Sie sollen ja deutlich von numerischen Variablen zu unterscheiden sein. Sie werden mit einem $ versehen.

Richtig, das ist das Symbol für den amerikanischen Dollar. Steht dieses Zeichen hinter einem Buchstaben, so bedeutet das für den Computer, daß der betreffende Buchstabe nun keine Zahl vertritt, sondern Buchstaben – also auch Wörter oder Sätze. Wir haben somit einen *Stellvertreter für Texte.*

Einen solchen aus Buchstaben und dem drangehängten Dollarzeichen zusammengesetzten Stellvertreter nennt man *String*-Variable. Mit "String" bezeichnet der Computerfachmann sogenannte "Zeichenketten". Zeichenketten können mehrere aneinandergereihte Buchstaben sein, also auch Wörter und Sätze. Dabei ist es dem Computer sogar egal, ob die Zeichenketten aus sinnvollem Text bestehen oder aus völlig wirrem Buchstabensalat.

Bei der Variablen für Zahlenwerte, also bei der numerischen Variablen, ordnen wir den Zahlenwert mit Hilfe des Gleichheitszeichens (=) dem Stellvertreter zu. Nicht anders ist es bei String-Variablen: Auch hier verwenden wir für die Zuordnung das Gleichheitszeichen. Der Text, den die String-Variable vertreten soll, steht immer in Anführungszeichen.

Wer's gleich probiert, hat's bald kapiert: Tippe

A$ = "MAX"

und

PRINT A$

Nachdem du RETURN gedrückt hast, steht auf dem Bildschirm in der nächsten Zeile:

MAX

Strings, also Zeichenketten, kannst du auch in Verbindung mit INPUT verwenden. Du tippst:

```
10 INPUT A$

20 PRINT A$

RUN
```

Der Computer fragt nach dem Input:

?

Du machst deine Eingabe:

MAX

Der Computer verarbeitet die Eingabe, und auf dem Bildschirm erscheint:

MAX

Alles klar?

String-Variablen lassen sich auch in Verbindung mit PRINT-Zeilen verwenden:

```
10 INPUT "WIE HEISST DU";A$

20 PRINT A$;" IST EIN SCHOENER NAME!"
```

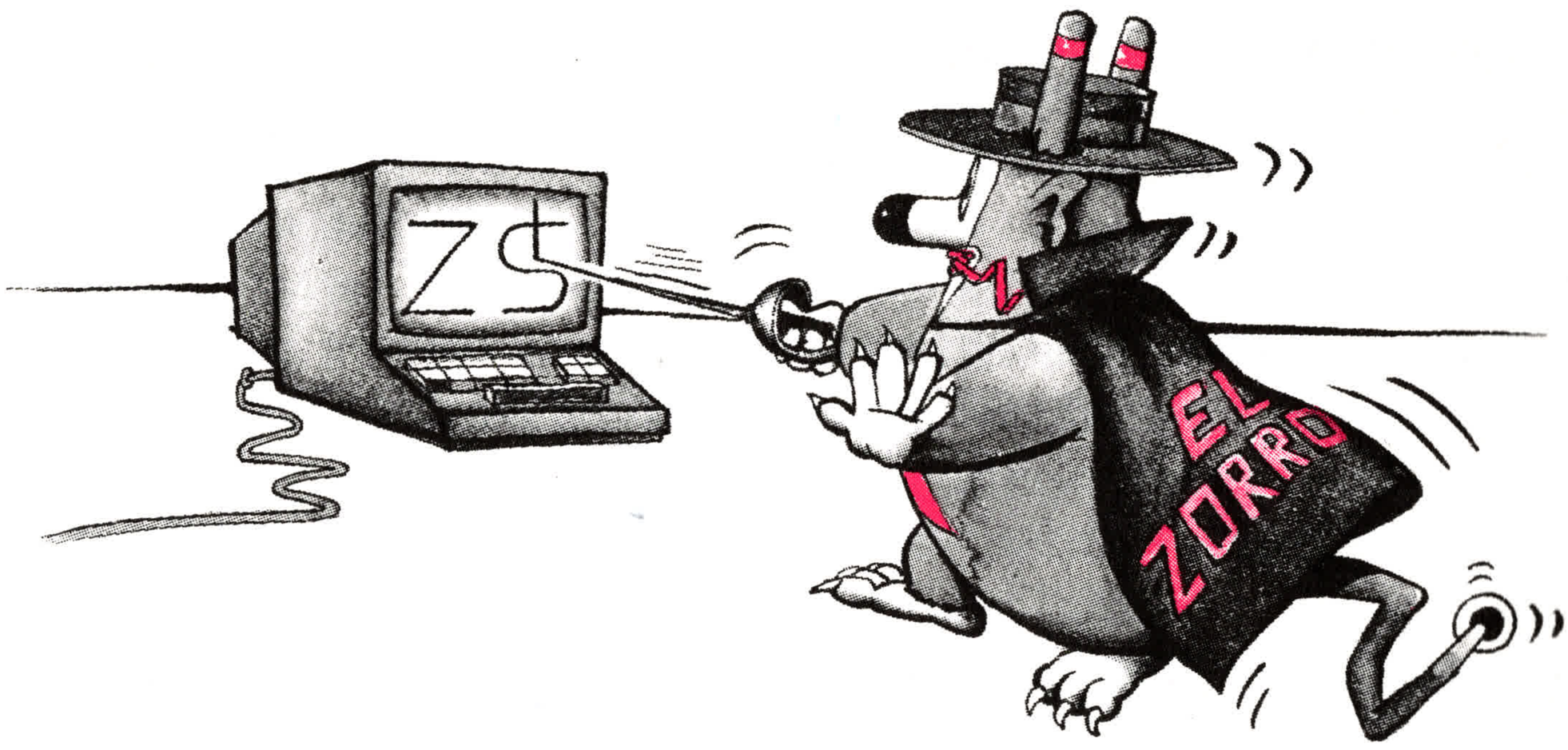
Auf folgende Weise verbindest du String-Variable und numerische Variable:

```
10 INPUT 'WIE HEISST DU BITTE?";A$
20 INPUT "UND WIE ALT BIST DU?";A
30 PRINT
40 PRINT "SO SO,":PRINT A$;" ,  DU BIST SCHON  ";A;" JAHRE ALT."
```

Mit den String-Variablen und dem Pluszeichen (+) kannst du aus einzelnen Wörtern und Satzteilen nach dem Baukastenprinzip Wörter zu längeren Texten zusammenfügen:

```
10 A$ = "COMPUTER-"

20 B$ = "MAX"

30 C$ = A$+B$

40 PRINT C$
```

Schon erkannt, was bei diesem Programm herauskommt? Prima!



Bei solchen Verknüpfungen mußt du genauestens auf Satzzeichen und Zwischenräume (Leerstellen) achten. Nur dann erhältst du gut lesbare Texte. Hier kannst du auch anhand von Leerzeilen für bessere Übersichtlichkeit sorgen – du weißt schon, mit PRINT ohne was dahinter.

Daß sich Strings wie Bausteine zusammenfügen lassen, ist vor allem bei der Arbeit mit dem Commodore 64 ein Segen. Er kann nämlich, wie schon bei der PRINT-Anweisung, nur eine bestimmte Textmenge innerhalb eines Strings vertragen. Werden die Texte zu lang, müssen wir sie in mehrere Strings aufteilen und diese dann aneinanderreihen.

Statt die einzelnen Strings mit Hilfe des Additionszeichens (+) zu verknüpfen, kannst du sie ebensogut hintereinander "ausdrucken" lassen:

```
10 A$ = "MAX"

20 B$ = "TEUFEL"

30 C$ = "PAPAGEI"

40 PRINT A$;B$;C$
```

Was würde passieren, wenn wir die Zeile 40 folgendermaßen schrieben:

```
40 PRINT A$,B$,C$,
```

Oder gar so:

```
40 PRINT A$:PRINT B$:PRINT C$
```

Du siehst, auf dem Bildschirm tut sich was . . . Jedenfalls bietet uns die Möglichkeit, Wörter und Satzteile als Input einzugeben ein neues Programmiermittel. Endlich können wir den Computer dazu bringen, mehr oder weniger sinnvolle Sätze zu bilden. In den Abbildungen HC 39 und HC 40 findest du einige Beispiele.

```
10 INPUT "WER";A$
20 INPUT "MACHT WAS";B$
30 INPUT "WIE";C$
40 INPUT "MIT DEM";D$
```

Was dabei herauskommen kann, wenn man daraus ein Spiel mit mehreren Mitspielern macht:

```
MIT DEM? KNALLKOPF
DER PRAESIDENT RUDERT ENERGISCH  MIT DEM
 KNALLKOPF.

READY.


MIT DEM? ZEITUNGSSTAENDER
HERR MEIER FASELT STUNDENLANG MIT DEM ZE
ITUNGSSTAENDER.

READY.


MIT DEM? TEUFEL
MAX TANZT BOOGIE WOOGIE-MAESSIG MIT DEM
TEUFEL.

READY.
```

Abb. HC 39

```
10  INPUT "WER";A$
20  INPUT "WUND WER NOCH";B$
30  INPUT "WAS";C$
40  INPUT "WUND WAS NOCH";D$
50  PRINT A$;" UND ";B$;" GINGEN"
60  PRINT "EINMAL SPAZIEREN."
70  PRINT"PLOETZLICH SAHEN SIE"
80  PRINT "EIN/E/M ";C$
90  PRINT "WO EIN/E ";C$;" IST,"
100 PRINT "SAGTE ";A$;" KAMM AUCH"
110 PRINT "EIN/E ";D$;" NICHT WEIT
120 PRINT "SEIN."
130 PRINT "ABER ";B$;" SAGTE:"
140 PRINT "WAS HAT DENN EIN/E ";C$
150 PRINT "MIT EINEM/EINER ";D$
160 PRINT "ZU TUN."
170 PRINT A$;" UND ";B$
180 PRINT "MUSSTEN LAUT LACHEN."
```

Noch ein Quasselprogramm und was dabei herauskommen kann:

```
UND WAS NOCH? SCHWEISSENKEL
DER TEUFEL UND MAX GINGEN
EINMAL SPAZIEREN.
PLOETZLICH SAHEN SIE
EIN/E/M COMPUTER
WO EIN/E COMPUTER IST,
SAGTE DER TEUFEL KAMM AUCH
EIN/E SCHWEISSENKEL NICHT WEIT
SEIN.
ABER MAX SAGTE:
WAS HAT DENN EIN/E COMPUTER
MIT EINEM/EINER SCHWEISSENKEL
ZU TUN.
DER TEUFEL UND MAX
MUSSTEN LAUT LACHEN.

READY.
```

Abb. HC 40

Anhand solcher und ähnlicher Programme kannst du nun ein wenig üben, bis du im Umgang mit der String-Verarbeitung etwas sicherer geworden bist.

In den Programmen aus den vorherigen Kapiteln konnte der Benutzer keinen Text als Antwort eingeben, sondern nur Ziffern. Ändere doch diese Programme so ab, daß man jetzt als Benutzer die Wörter "JA" und "NEIN" eingeben kann. In Abbildung

HC 41 zeigen wir hierzu als Übungsbeispiel eine geänderte Fassung unseres Umrechnungsprogramms.

```
10 REM * ZOLL UND CM *
20 PRINT "␣GUTEN TAG."
30 PRINT:PRINT "WOLLEN SIE CM IN ZOLL UMRECHNEN?"
40 PRINT "DANN GEBEN SIE >CM/ZOLL< EIN."
50 PRINT:PRINT "ODER WOLLEN SIE ZOLL IN CM UMRÈCHNEN?"
60 PRINT "DANN GEBEN SIE >ZOLL/CM< EIN."
70 INPUT A$
80 IF A$="CM/ZOLL" THEN GOTO 110
90 IF A$="ZOLL/CM" THEN GOTO 140
100 PRINT "DIESE AMTWORT VERSTEHE ICH NICHT!":GOTO 30
110 PRINT "WIEVIEL CM"
120 INPUT B
130 PRINT B;" CM SIND ";B*0.38;" ZOLL.":GOTO 170
140 PRINT "WIEVIEL ZOLL"
150 INPUT C
160 PRINT C;" ZOLL SIND ";C*2.6;" CM."
170 PRINT "NOCH EINE UMRECHNUMG?"
180 PRINT "DANN GEBEN SIE BITTE >JA< EIN."
190 INPUT D$
200 IF D$="JA" THEN GOTO 30
210 PRINT "AUF WIEDERSEHEN!":END
```

Abb. HC 41

Bestimmt erinnerst du dich noch an unser KESSA-Prinzip. Jedes KESSA-Programm könnte nun damit beginnen, daß der Computer nach dem Namen des Benutzers fragt und ihn dann namentlich begrüßt.

Auch die Antworten des Computers ließen sich freundlicher gestalten, indem er den Benutzer hin und wieder persönlich anredet.

Nehmen wir als Beispiel ein Rateprogramm. Zwei Fassungen davon findest du in den Abbildungen HC 42 und HC 43.

Was hältst du von dem Gedanken, Vornamen, Familiennamen, Adressen und Telefonnummern aller deiner Klassenkameraden zu "verstringen"? Bei so einem Programm kommt es sehr darauf an, daß du die String-Variablen streng systematisch benennst.

Wie die numerische Variable kann auch die String-Variable aus zwei Zeichen bestehen. Das String-Symbol ($) kommt außerdem extra dazu. Das erste Zeichen muß stets ein Buchstabe sein. Als zweites Zeichen kannst du auch eine Zahl verwenden. Also beispielsweise:

A1$

```
1 INPUT "WIE HEISST DU";NA$
2 PRINT "MACHEN WIR EIN KLEINES RATESPIEL, ";NA$
10 PRINT "RATE EINE ZAHL"
20 PRINT "ZWISCHEN 1 UND 10 ";NA$
30 INPUT A
40 IF A=3 THEN GOTO 70
50 PRINT "TUT MIR LEID, "NA$;" NOCH EIN VERSUCH!": GOTO 30
70 PRINT "HERZLICHEN GLUECKWUNSCH, ";NA$
80 PRINT "DIE 3 WAR GESUCHT.":END

READY.
```

Abb. HC 42

```
10 INPUT "WIE HEISST DU";NA$
20 PRINT "MACHEN WIR EIN KLEINES RATESPIEL, ";NA$;"!"
30 PRINT "RATE EINE ZAHL"
40 PRINT "ZWISCHEN 1 UND 10,";NA$;"!"
50 INPUT A
60 IF A=3 THEN GOTO 80
70 PRINT "TUT MIR LEID, "NA$;", NOCH EIN VERSUCH!": GOTO 30
80 PRINT "HERZLICHEN GLUECKWUNSCH, ";NA$;","
90 PRINT "DIE 3 WAR GESUCHT.":END

READY.
```

Abb. HC 43

Welche der folgenden Variablen lassen sich sinnvoll als Stellvertreter für Zahlen verwenden – das heißt als *numerische Variablen?*

A3	6T	UU	MAX	MUENCHEN	PAPAGEI
= 6	AA	PO	!Q	??	

Und welche von folgenden Variablen taugen für Strings?

AA$ A&$ 65$ AE$ K5$ TEUFEL$?I$ Z$

Die Lösung geben wir hier absichtlich nicht an. Bei einigen der angebotenen Variablennamen dürfte dir zweifellos sofort klar sein, ob sie sich verwenden lassen. Bei anderen wirst du es wohl erst ausprobieren müssen.

Zurück zu unserer Idee, die Namen deiner Schulkameraden zu verstringen. Man könnte doch ein Programm schreiben, das die Namen deiner Mitschüler ihrem jeweiligen Sitzplatz im Klassenzimmer zuordnet.

Dazu skizzierst du zweckmäßigerweise einen Klassensitzplan. Der könnte so aussehen wie in Abbildung Z 22.

A		B		C	
1	2	1	2	1	2
Claudia	Michaela	Peter	Tom	Traugott	Hans-Jürgen
3	4	3	4	3	4
Sarah	Felicita	Michel	Yilmaz	Patrizia	Monika
5	6	5	6	5	6
Franz K.	Max	Mehmet	Fritz	Norbert	(ich selber)
7		7	8	7	8
Kora		Franz M.	Veronika	Carlo	Otto

Abb. Z 22

Der Buchstabe bezeichnet die Bankreihe, die Zahl den Sitzplatz in der jeweiligen Reihe. So kennzeichnet beispielsweise B 3 den 3. Platz in der mittleren Bankreihe. Unser Programm soll

nun nach Eingabe des Platzes den Namen des darauf sitzenden Schülers ausgeben.

Versuchen wir nun eine Beschreibung des Programms zu erstellen. Wir stellen es uns so vor:

Programmbeschreibung

Zuerst soll auf dem Bildschirm eine Grafik erscheinen. Darauf sind die einzelnen Sitzreihen und die Sitzplätze schematisch dargestellt. Die zur Darstellung geeigneten Zeichen suchst du auf dem Commodore 64 nach deinen Vorstellungen heraus.

Anschließend soll der Benutzer zwei Eingaben machen:

1. Die Bankreihe, in welcher der Schüler sitzt.
2. Den Platz in dieser Reihe, auf dem der gesuchte Schüler sitzt.

Die erste Eingabe ist ein Buchstabe, die zweite besteht aus einer Zahl. Beide Eingaben sollen zu einem String zusammengefaßt werden. Also lassen wir den Benutzer auch die Zahl als String eingeben.

Im nächsten Schritt soll das Programm jedem Sitzplatz in jeder Reihe den Namen des dort sitzenden Schülers als String zuordnen. Der Variablenname des Strings soll mit der Bezeichnung des Sitzplatzes übereinstimmen.

Als Output antwortet das Programm mit folgender Bildschirmmeldung:

”IN DER REIHE ___ SITZT AUF PLATZ ___ _____“

Das Programm muß jetzt die Eingaben verarbeiten und die eingegebenen Daten den ”gespeicherten“ Namen zuordnen. So viele Schüler in der Klasse sitzen so viele IF/THEN-Zeilen benötigen wir dazu – jedenfalls mit unseren bisherigen BASIC-Programmierkenntnissen.

Außerdem benötigt das Programm einen Schutz gegen unsinnige Eingaben. Gibt der Anwender zum Beispiel ”MISTHAUFEN“ ein, soll der Computer antworten, daß auf diesem Platz niemand sitzt, weil es diesen Platz nicht gibt. Das ist hier gar nicht schwierig.

Alle sinnvollen Eingaben sind ja bereits vollständig in Form der IF/THEN-Zeilen aufgestellt. Findet der Computer eine Eingabe nicht, nachdem er all diese Zeilen abgesucht hat, ”erkennt“ er, daß diese Eingabe sinnlos war. Daraufhin lassen

wir ihn mit einer entsprechenden Bildschirmmeldung antworten.

Zum Schluß bauen wir noch eine Endabfrage ein: Will der Anwender noch einen weiteren Schülernamen wissen?

- Wenn *ja,* soll das Programm noch mal von vorn beginnen.
- Wenn *nicht,* soll sich das Programm verabschieden – höflich wie immer.

Diese Beschreibung gibt uns schon eine ungefähre Vorstellung davon, wie das Programm aussehen wird und welche BASIC-Begriffe wir einsetzen müssen.

Programmablaufplan

Jetzt entwerfen wir das Flußdiagramm. Als Leitfaden dient uns die Programmbeschreibung.

In den Abbildungen (Z 23, Z 24 und Z 25) siehst du, wie wir das Flußdiagramm Schritt für Schritt aufbauen und vervollständigen.

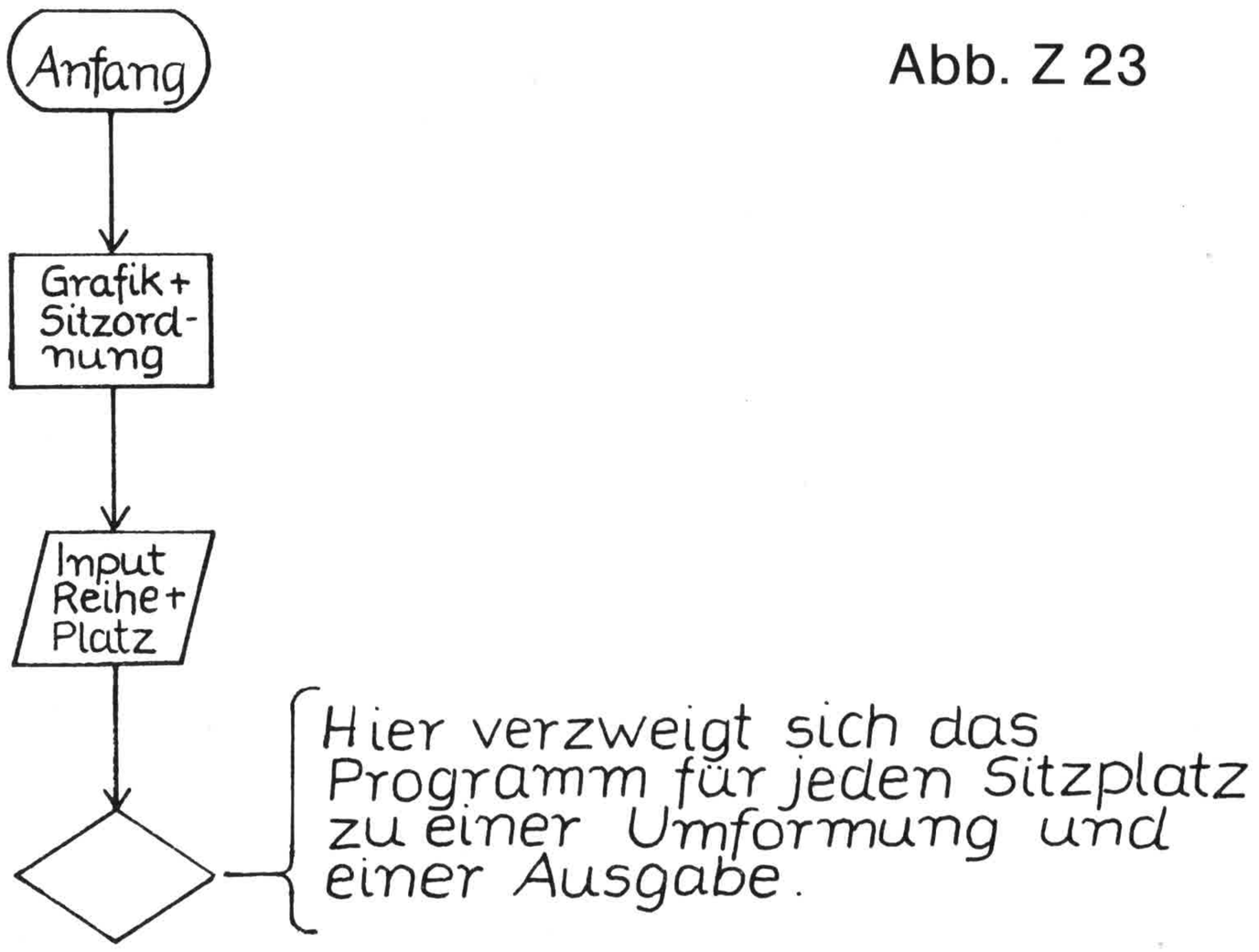


Abb. Z 23

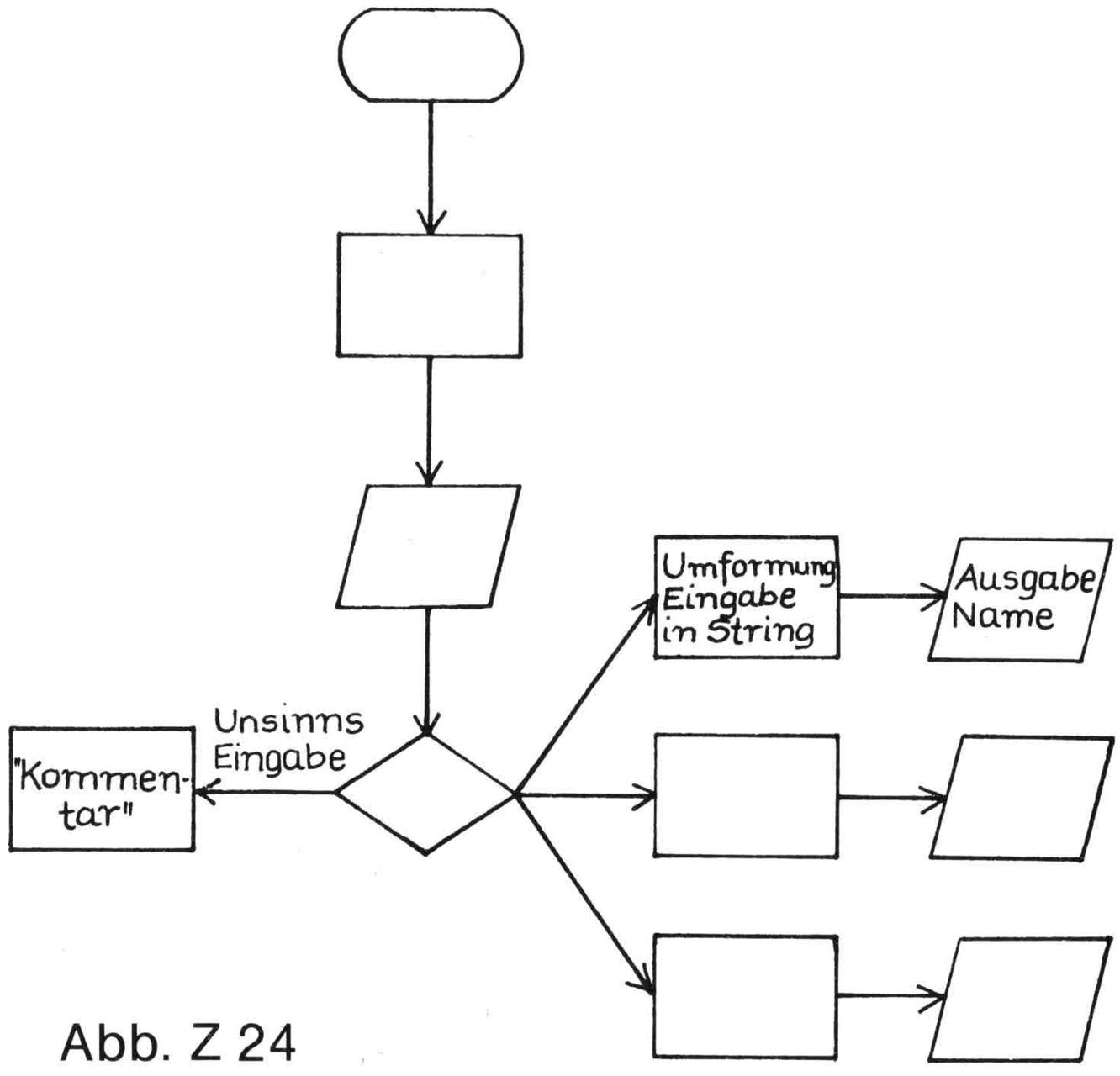


Abb. Z 24

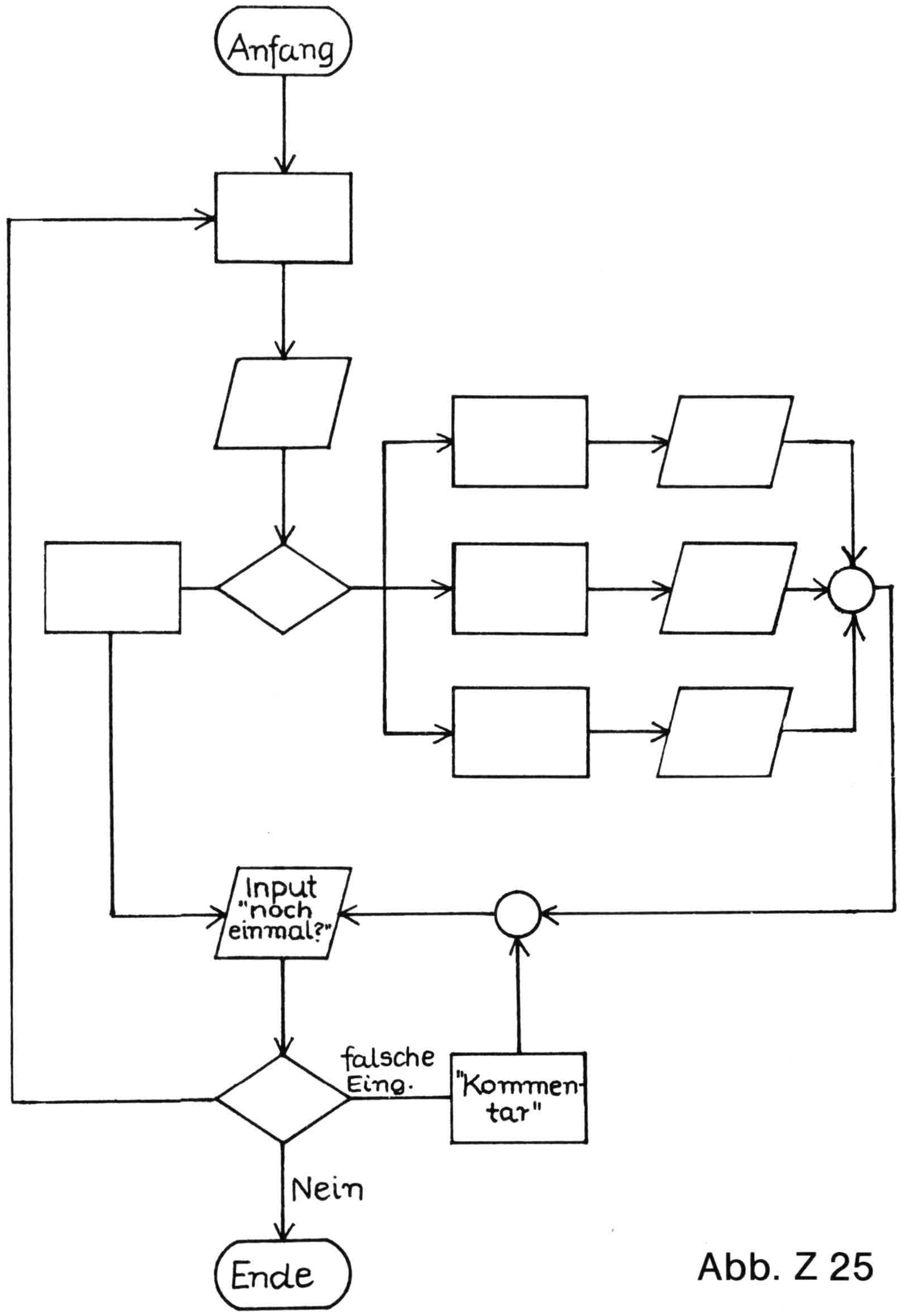


Abb. Z 25

Das fertige Flußdiagramm könnte so aussehen, wie in Abbildung Z 26.

Variablenliste

Die Variablenliste zu diesem Programm ist bereits ziemlich umfangreich. Zum Glück haben wir durch den Sitzplan die String-Variablen schon geordnet.

Codieren und Eintippen

Halte dich nun bei der Umsetzung in die entsprechenden BASIC-Programmzeilen bitte genau an das vorliegende Fluß-

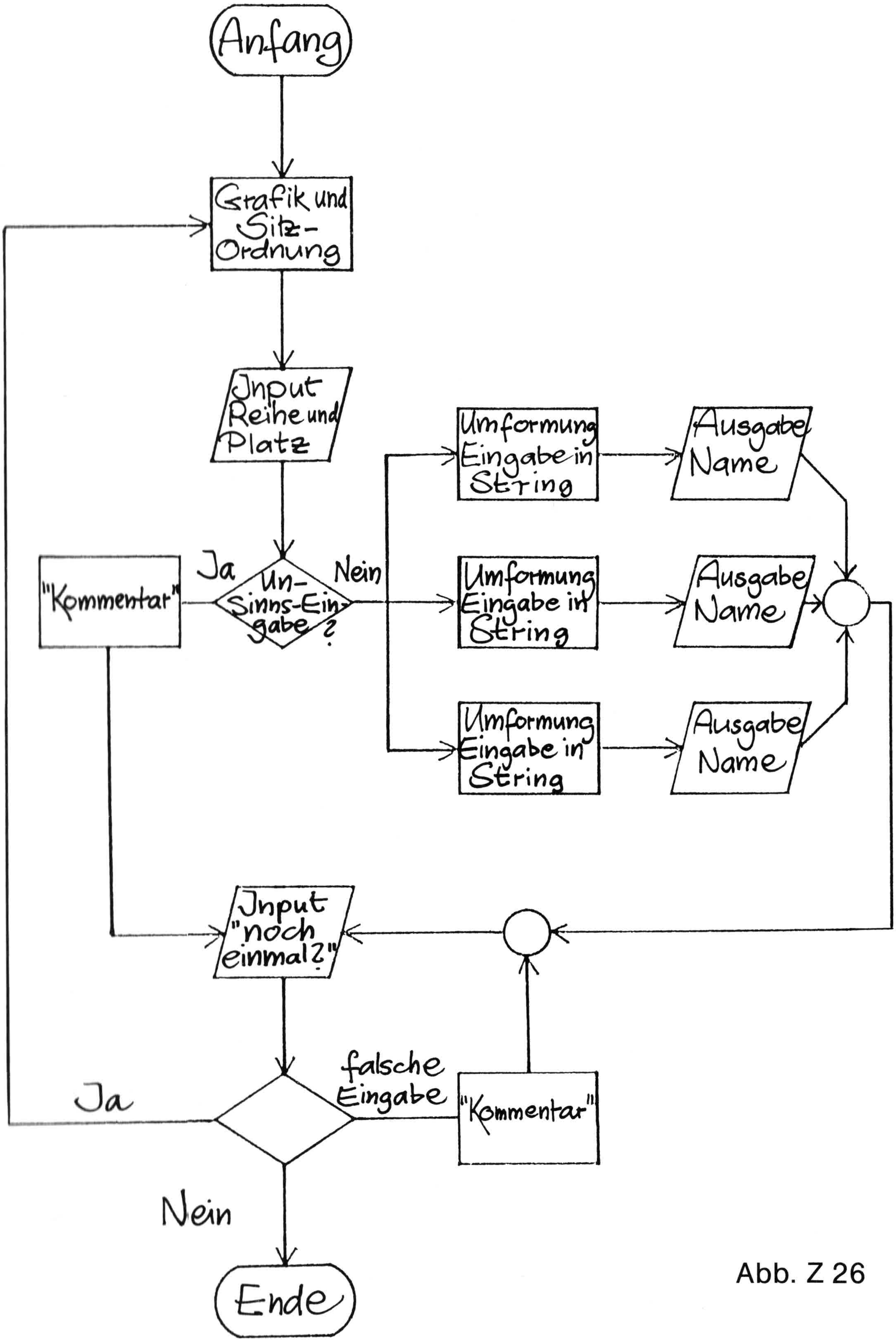


Abb. Z 26

diagramm. Anschließend hast du immer noch Gelegenheit, das eingegebene und *durchgetestete* Programm nach deinen Vorstellungen abzuwandeln. Und die *Dokumentation* nicht vergessen!

Unsere fertige Programmversion siehst du in den Abbildungen HC 44 bis HC 46 (Seiten 113–117).

Alte Hasen würden dir jetzt sagen, daß man solch ein Programm mit einigen Tricks in eine noch kürzere Form bringen kann. Dazu fehlt uns im Augenblick noch das nötige Handwerkszeug, nämlich die entsprechenden BASIC-Begriffe. Diese neuen Befehle werden wir in Band 5 (BASIC für Kenner) vorstellen.

Hauptsache ist, auch dieses Programm funktioniert einwandfrei. Vor allem darfst du wirklich stolz darauf sein, falls dir das gelungen ist, ohne vorher viel in unseren Vorschlägen gespickt zu haben. Das bedeutet nämlich, daß du auf dem Gebiet der "Computer-Logik" mittlerweile ein großes Stück weitergekommen bist.

"Computerliebe"

Abschließend wollen wir noch ein typisches "Dialogprogramm" vorstellen. Auf dieses Programm bezieht sich die Kapitelüberschrift. Da macht der Computer dem Benutzer eine Liebeserklärung, nachdem er aus etlichen Eingaben seine Schlüsse gezogen hat. Einige unserer Testleser kamen zu dem Eindruck, daß es schon gewisse Ähnlichkeit mit ELIZA hat – du weißt, mit dem Seelenmassage-Programm des amerikanischen Professors.

Dieses Dialogprogramm ist, wie du im Listing erkennen wirst, zwar sehr lang. Dennoch hat es im Grunde einen sehr einfachen Aufbau. Gewiß wird es dir kaum schwerfallen, es "auseinanderzunehmen", also in eine Programmbeschreibung umzusetzen und als Flußdiagramm darzustellen. Dann reizt es dich möglicherweise auch, dieses Programm abzuwandeln und vielleicht ganz anders wieder zusammenzusetzen oder gar neue Elemente einzufügen.

Die allergrößte Sorgfalt erfordert hier die Variablenliste. Denn an Variablen hat dieses Programm eine Menge zu bieten. Die Variablen der "sinnvollen" Eingaben haben wir entsprechend ihrem Zweck benannt. So ist zum Beispiel NA$ die String-Variable für den *Na*men des Benutzers.

Die Liste enthält außerdem noch die Variablen für "unsinnige" Eingaben (wie zum Beispiel in den Zeilen 240, 260 und

```
10 PRINT "CLKLASSE 5A"
20 PRINT:PRINT
30 PRINT,"A","B","C"
40 PRINT,"┌─────┐","┌─────┐","┌─────┐"
50 PRINT,"│ 1   2 │","│ 1   2 │","│ 1   2 │"
60 PRINT,"│ 3   4 │","│ 3   4 │","│ 3   4 │"
70 PRINT,"│ 5   6 │","│ 5   6 │","│ 5   6 │"
80 PRINT,"│ 7     │","│ 7   8 │","│ 7   8 │"
90 PRINT,"└─────┘","└─────┘","└─────┘"
100 INPUT "CLCLGIB DEN BUCHSTABEN FUER DIE REIHE EIN";A$
110 INPUT "CLGIB DIE ZAHL FUER DEN SITZPLATZ EIN";B$
120 C$=A$+B$
130 A1$="CLAUDIA":A2$="MICHAELA":A3$="SARAH":A4$="FELICITA"
140 A5$="FRANZ K.":A6$="MAX":A7$="KORA"
150 B1$="PETER":B2$="TOM":B3$="MICHEL":B4$="YILMAZ":B5$="MEHMET"
160 B6$="FRITZ":B7$="FRANZ M.":B8$="VERONIKA"
170 C1$="TRAUGOTT":C2$="HANS-JUERGEN":C3$="PATRIZIA":C4$="MONIKA"
180 C5$="NORBERT":C6$="NIEMAND ANDERS ALS ICH SELBER"
190 C7$="CARLO":C8$="OTTO"
```

I.

II.

III.

I. Zeile 10 bis 90:
   Die Bildschirmgrafik
II. Zeile 100 bis 120:
   Die Eingabe und die Verbindung der beiden Strings
III. Zeile 130 bis 190:
   Die ”Speicherung“ der Sitzordnung

Abb. HC 44

```
10 PRINT "■KLASSE 5A"
20 PRINT:PRINT
30 PRINT,"A","B","C"
40 PRINT,"┌──────┐","┌──────┐","┌──────┐"
50 PRINT,"│ 1  2 │","│ 1  2 │","│ 1  2 │"
60 PRINT,"│ 3  4 │","│ 3  4 │","│ 3  4 │"
70 PRINT,"│ 5  6 │","│ 5  6 │","│ 5  6 │"
80 PRINT,"│ 7    │","│ 7  8 │","│ 7  8 │"
90 PRINT,"└──────┘","└──────┘","└──────┘"
100 INPUT "■■GIB DEM BUCHSTABEN FUER DIE REIHE EIN";A$
110 INPUT "■■GIB DIE ZAHL FUER DEM SITZPLATZ EIN";B$
120 C$=A$+B$
130 A1$="CLAUDIA":A2$="MICHAELA":A3$="SARAH":A4$="FELICITA"
140 A5$="FRANZ K.":A6$="MAX":A7$="KORA"
150 B1$="PETER":B2$="TOM":B3$="MICHEL":B4$="YILMAZ":B5$="MEHMET"
160 B6$="FRITZ":B7$="FRANZ M.":B8$="VERONIKA"
170 C1$="TRAUGOTT":C2$="HANS-JUERGEN":C3$="PATRIZIA":C4$="MONIKA"
180 C5$="NORBERT":C6$="NIEMAND ANDERS ALS ICH SELBER"
190 C7$="CARLO":C8$="OTTO"
200 PRINT "■■IN DER REIHE ";A$;" SITZT AUF PLATZ ";B$;" "
210 IF C$="A1"THEN PRINT A1$:GOTO 470
220 IF C$="A2"THEN PRINT A2$:GOTO 470
230 IF C$="A3"THEN PRINT A3$:GOTO 470
240 IF C$="A4"THEN PRINT A4$:GOTO 470
250 IF C$="A5"THEN PRINT A5$:GOTO 470
260 IF C$="A6"THEN PRINT A6$:GOTO 470
270 IF C$="A7"THEN PRINT A7$:GOTO 470
280 IF C$="B1"THEN PRINT B1$:GOTO 470
290 IF C$="B2"THEN PRINT B2$:GOTO 470
300 IF C$="B3"THEN PRINT B3$:GOTO 470
```

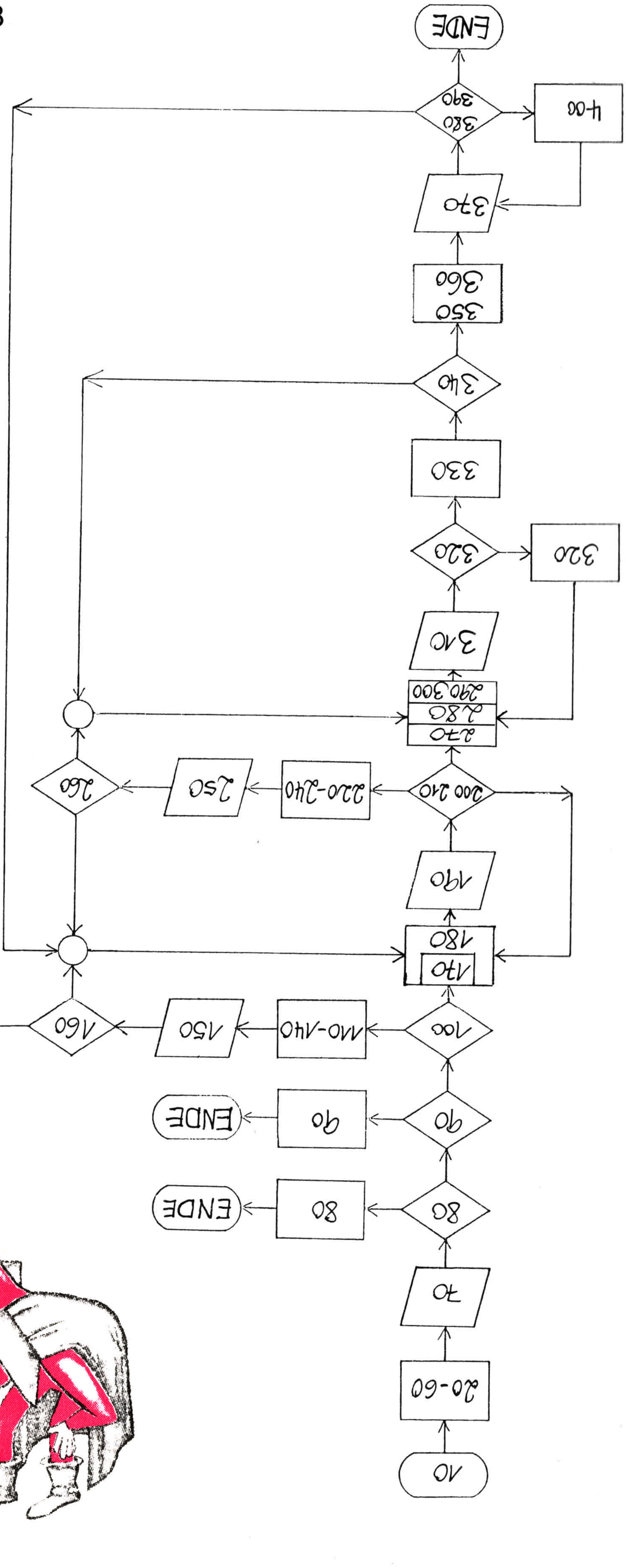
```
310 IF C$="B4"THEN PRINT B4$:GOTO 470
320 IF C$="B5"THEN PRINT B5$:GOTO 470
330 IF C$="B6"THEN PRINT B6$:GOTO 470
340 IF C$="B7"THEN PRINT B7$:GOTO 470
350 IF C$="B8"THEN PRINT B8$:GOTO 470
360 IF C$="C1"THEN PRINT C1$:GOTO 470
370 IF C$="C2"THEN PRINT C2$:GOTO 470
380 IF C$="C3"THEN PRINT C3$:GOTO 470
390 IF C$="C4"THEN PRINT C4$:GOTO 470
400 IF C$="C5"THEN PRINT C5$:GOTO 470
410 IF C$="C6"THEN PRINT C6$:GOTO 470
420 IF C$="C7"THEN PRINT C7$:GOTO 470
430 IF C$="C8"THEN PRINT C8$:GOTO 470
440 PRINT "NIEMAND! DEN PLATZ "
450 PRINT "GIBT ES GAR NICHT IN"
460 PRINT "UNSERER KLASSE!"
470 PRINT "□□MOECHTEST DU NOCH ETWAS WISSEN?"
480 INPUT "TIPPE >JA< ODER >NEIN< EIN";D$
490 IF D$="JA"THEN PRINT"□":GOTO 10
500 IF D$="NEIN" THEN PRINT"□□BIS ZUM NAECHSTEN MAL":END
510 PRINT "VERSTEHE ICH NICHT!":GOTO 480

READY.
```

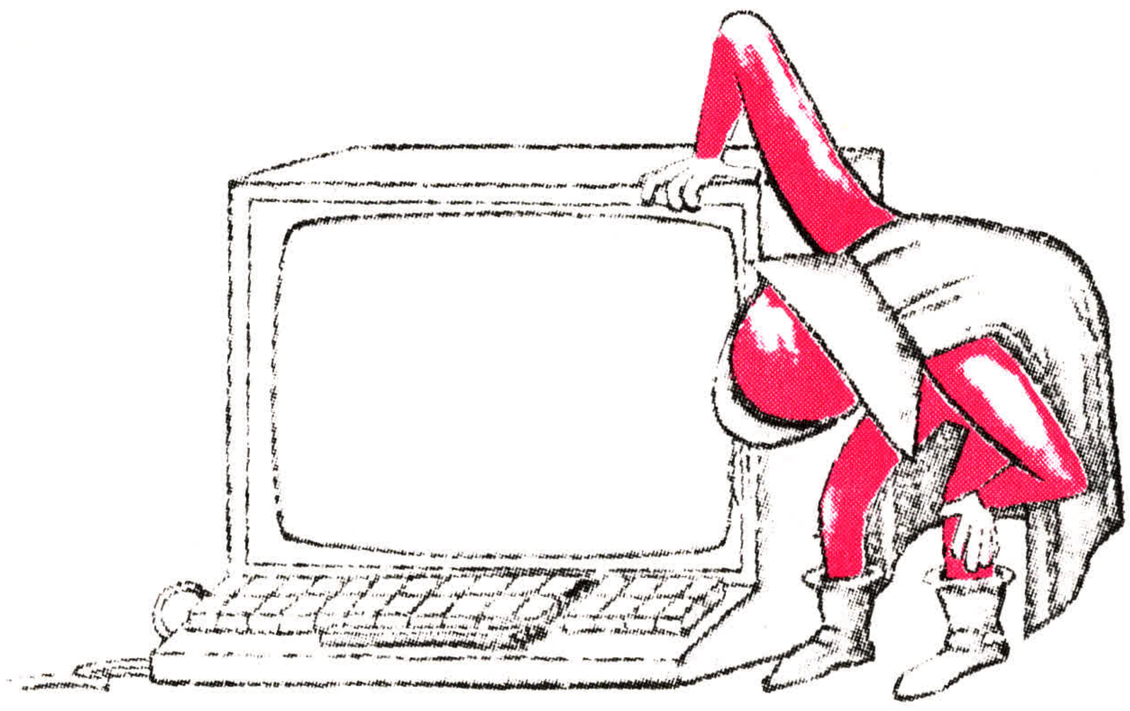
VII.

VII. Zeile 470 bis 510:
Endabfrage nach Wiederholung

Abb. HC 46

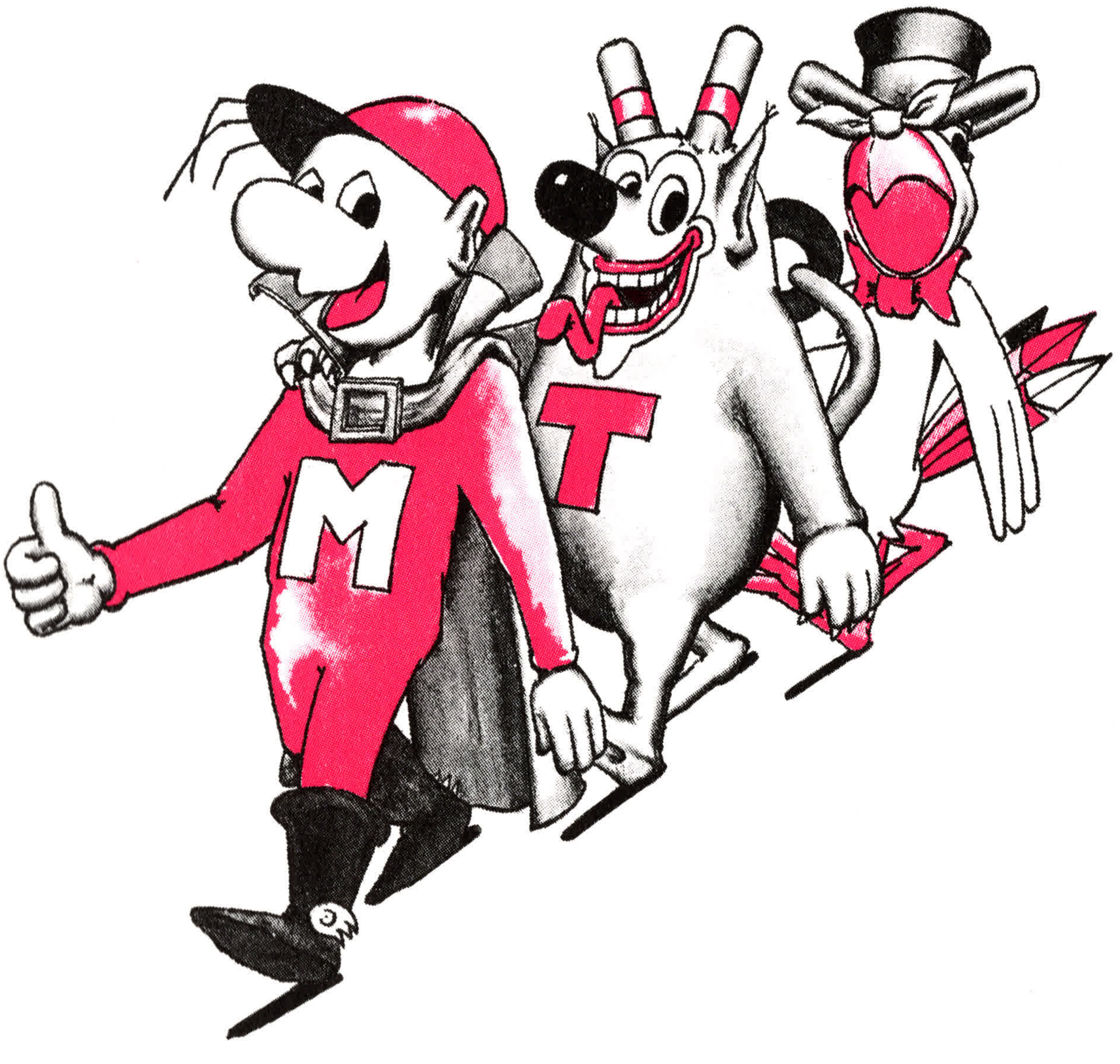
Auflösung von Seite 98





280). Bei der Namensvergabe für diese Variablen als auch für die numerischen Variablen sind wir einfach nach dem Alphabet vorgegangen.

Wie das Einmaleins-Programm vom COMPUTERMAX und das Programm "Sitzplan" ist auch die elektronische Liebeserklärung auf der Diskette zu diesem Buch enthalten. Falls du also schon das Buch mit Diskette haben solltest, kannst du getrost herumexperimentieren. Zuvor empfiehlt es sich, von der Diskette eine "backup" anzulegen. Dann kannst du die Programme ohne Sorge vor Fehlschlägen nach deinen Wünschen verändern – denn der Programmierer bist jetzt du!



Was heißt was in BASIC

Die wichtigsten BASIC-Begriffe aus diesem Buch

So heißt es in BASIC:	**Und das soll der Computer tun:**
AND	Lasse bei IF . . . THEN mehrere Bedingungen gelten.

Beispiel:

```
10 INPUT A
20 INPUT B
30 PRINT A+B
40 A=A+1
50 B=B+2
60 IF A>10 AND B>20THEN END
70 GOTO 30
```

END	Beende das Programm
GOTO	Gehe zur Zeile, deren Nummer nach GOTO kommt

Beispiel:

```
10 PRINT "HALLO"
20 PRINT "COMPUTER"
30 GOTO 10
```

IF . . . THEN	**WENN** eine Bedingung erfüllt ist, **DANN** springe zur Zeile, die nach THEN steht

Beispiel:

```
10 INPUT A
20 IF A>10 THEN GOTO 10
30 IF A<10 THEN PRINT "NA ENDLICH"
40 END
```

INPUT	Frage erst nach einer Eingabe (zum Beispiel einer Zahl oder einem Buchstaben), bevor du ein Programm weiter bearbeitest.

Beispiel:

```
10 INPUT A
20 PRINT "A*10=";A*10
30 GOTO 10
40 END
```

LIST	Zeige das Programm-Listing auf dem Bildschirm.

LOAD	Lade ein Programm vom Datenträger (Kassette oder Diskette)

Beispiel:

LOAD "COMPUTERLIEBE" lädt ein Programm mit dem Namen "COMPUTERLIEBE" von der Kassette. Auf der Datasette oder auf dem Kassettenrecorder wird PLAY gedrückt.

LOAD "COMPUTERLIEBE", 8 lädt das Programm "COMPUTER-LIEBE" von der Diskettenstation.

NEW — Lösche dein Gedächtnis.

OR — Lasse bei IF . . . THEN mehrere Bedingungen gelten.

Beispiel:

```
10 INPUT A
20 INPUT B
30 PRINT A+B
40 A=A+1
50 B=B+2
60 IF A=10 OR B>20THEN END
70 GOTO 30
```

PRINT — **mit Anführungszeichen:** Schreibe oder zeichne auf den Bildschirm, was zwischen den Anführungszeichen steht.

Beispiel:

```
10 PRINT"MAX"
```

PRINT — **ohne Anführungszeichen:** Führe aus, rechne aus, was hinter dem PRINT steht.

Beispiel:

```
10 PRINT 23-9
```

REM — Schreibe eine Bemerkung des Programmierers in das Listing. Sie soll das Programm selbst nicht beeinflussen.

Beispiel:

```
10 REM *ZWEIER-EINMALEINS*
20 INPUT A
30 PRINT A*2
40 GOTO 20
```

RETURN

Funktionstaste:
Merke dir, was ich gerade geschrieben habe und gehe mit dem Cursor an den Anfang der nächsten Zeile.

RUN

Führe das Programm aus.

RUN / STOP

Funktionstaste:
Halte das Programm dort an, wo du gerade bist und melde mir mit BREAK IN . . . (Zeilennummer), wo die Unterbrechung stattgefunden hat.

SAVE

Speichere ein Programm auf dem Datenträger (Kassette oder Diskette) ab.

Beispiel:

SAVE "COMPUTERLIEBE“ speichert ein Programm auf Kassette.

Es müssen die beiden Tasten PLAY und RECORD gedrückt werden.

SAVE "COMPUTERLIEBE“, 8 speichert das Programm "COMPUTERLIEBE“ auf Diskette.

Tips und Tricks für das Eingeben der Listings

Beim Abtippen der Listings schleichen sich gern Fehler ein. Die folgenden Tips sollen dich davor bewahren und helfen, Fehler schnell zu finden.

TIP 1
Lege das Listing, das du eintippen willst, so vor dich hin, daß du es gut lesen kannst. Als Lesehilfe kannst du ein unbeschriebenes Blatt verwenden. Lege es genau unter die Zeile, die du als nächstes eingeben willst.

TIP 2
Hake mit einem Stift jede eingegebene Zeile ab. So gibst du keine Zeile doppelt ein.

TIP 3
Speichere dein Programm des öfteren mal ab, zum Beispiel alle 10 Zeilen. Sollte inzwischen der Rechner abstürzen, gehen nur die Zeilen verloren, die du noch nicht abgespeichert hast.

TIP 4
Spätestens wenn du das eingegebene Programm starten willst, solltest du das Programm zuvor abspeichern. Denn eventuelle Fehler können den Rechner so blockieren, daß du nur durch Ausschalten des Rechners wieder in den normalen Betrieb gelangst. Das Programm wäre damit verloren. Sobald du das Programm abgespeichert hast, kannst du es wieder in den Rechner laden und den Fehler suchen.

TIP 5
Falls du mit BASIC schon einigermaßen vertraut bist, noch ein weiterer Tip: Wenn Verzweigungen kommen (GOTO, GOSUB), mach's so, wie der Rechner später: "Springe" in diese Verzweigungen und tippe diese zuerst ab, bevor du mit dem Hauptprogramm weitermachst. Das hat den Vorteil, daß du jetzt schon Probestarts des Programmes machen und schon etwaige Fehler ausbessern kannst. Beachte dabei TIP 4.

```
100 :        ----------------
101 :         STEUERZEICHEN
102 :        ----------------
103 :
104 :     1) CURSORSTEUERTASTEN
105 :    "■" CRSR              CURSOR RECHTS
106 :    "■" SHIFT CRSR        CURSOR LINKS
107 :    "■" CRSR              CURSOR RUNTER
108 :    "■" SHIFT CRSR        CURSOR HOCH
109 :    "■" CLRHOME           CURSOR INS LINKE ECK
110 :    "■" SHIFT CLRHOME     BILDSCHIRM LOESCHEN
111 :
112 :     2) FUNKTIONSTASTEN
113 :    "■" F1                FUNKTIONSTASTE 1
114 :    "■" SHIFT F1          FUNKTIONSTASTE 2
115 :    "■" F3                FUNKTIONSTASTE 3
116 :    "■" SHIFT F3          FUNKTIONSTASTE 4
117 :    "■" F5                FUNKTIONSTASTE 5
118 :    "■" SHIFT F5          FUNKTIONSTASTE 6
119 :    "■" F7                FUNKTIONSTASTE 7
120 :    "■" SHIFT F7          FUNKTIONSTASTE 8
121 :
122 :     3) FARBEN
123 :    "■" CTRL 1            SCHWARZ
124 :    "■" CTRL 2            WEISS
125 :    "■" CTRL 3            ROT
126 :    "■" CTRL 4            TUERKIS
127 :    "■" CTRL 5            VIOLETT
128 :    "■" CTRL 6            GRUEN
129 :    "■" CTRL 7            BLAU
130 :    "■" CTRL 8            GELB
131 :    "■" C= 1              ORANGE
132 :    "■" C= 2              BRAUN
133 :    "■" C= 3              HELLROT
134 :    "■" C= 4              GRAU 1
135 :    "■" C= 5              GRAU 2
136 :    "■" C= 6              HELLGRUEN
137 :    "■" C= 7              HELLBLAU
138 :    "■" C= 8              GRAU 3
139 :
140 :     4) SONSTIGE STEUERZEICHEN
141 :    "■" CTRL 9            REVERSSCHRIFT AN
142 :    "■" CTRL 0            REVERSSCHRIFT AUS
```

BİT
TEM

Listings der Spiele von Band 3

1. RENUMBER
2. VOKABELPAUKPROGRAMM
3. COMPUTERLIEBE

"Die in diesen Listings enthaltenen Programme sind aus Bausteinen zusammengesetzt. Einige dieser Bausteine sind zum einwandfreien Funktionieren nicht unbedingt nötig, sondern dienen nur als 'Bild' bzw. 'Grafik'. Jeder Programmbaustein ist durch REM-Zeilen gekennzeichnet. Beim Abtippen kann man sich also schon überlegen, ob es nicht einfacher ist, auf die entbehrlichen Programmbausteine zu verzichten! Das spart Zeit, und man hat auch weniger Tippfehler zu befürchten."

RENUMBER

Mit diesem Programm lassen sich selbstgeschriebene Programme neu numerieren. Du gehst folgendermaßen vor:

1. Das Programm, das du umnumerieren möchtest, speicherst du zuerst ab.
2. Du lädst das Programm "RENUMBER.ASM" und startest es mit RUN.
3. Das Programm, das man umnumerieren möchte, wird geladen.
4. Mit der Eingabe "SYS 49152, /Erste Zeile/,/Zeilenschritt/" wird das Programm umnumeriert.
   SYS 49152, 100,10 bedeutet also:
   die neue Numerierung des Programms beginnt mit Zeile 100.
   Das Programm wird in Zehnerschritten durchnumeriert.
5. Mit LIST kannst du die neue Numerierung überprüfen.

Achtung!
Gib niemals sinnlose Werte ein! Dies kann zur Programmzerstörung führen (z. B. bei Zeilenschritt = 0). Zu einer Fehlermeldung kommt es, wenn eine der neuen Zeilennummern über dem Wert 65536 liegt.
Empfohlene Werte für erste Zeile und Zeilenschritt:
erste Zeile: 0,10,100,1000,10000
Zeilenschritt: 10

Merkmale der beiden RENUMBER-Versionen

I. RENUMBER.DAT

Da unser Renumber-Programm in Maschinensprache geschrieben wurde, besteht diese Version vor allem aus DATA-Zeilen. Sie enthalten die Informationen für den Rechner. Mit diesen DATA-Zeilen wird es möglich, das Programm abzutippen. Die Version ist somit vor allem für diejenigen von Interesse, die dieses Buch ohne Diskette gekauft haben und die Programme selber eintippen (lassen).

Das Programm RENUMBER.DAT stellt im Rechner erst ein Maschinenprogramm her, das mit dem SYS-Befehl gestartet werden kann. (Siehe RENUMBER.ASM.)

Nachdem du das BASIC-Programm Renumber.DAT abgetippt hast, solltest du es erst abspeichern und dann starten. Erscheint auf dem Bildschirm die Meldung

– FEHLER in BLOCK –

hat man in dem angegebenen Block einen Tippfehler gemacht, der ausgebessert werden muß.

Sobald das BASIC-Programm ohne Fehler abgearbeitet wurde, ist ein Maschinenprogramm entstanden. Dies kannst du laden und speichern wie andere dir vertraute Programme. Es besteht allerdings aus nur noch einer einzigen BASIC-Zeile mit einem SYS-Befehl, der das Maschinenprogramm startet.

Das entstandene Maschinenprogramm speichert man am besten unter dem Namen RENUMBER.ASM ab.

II. RENUMBER.ASM

Dies ist die Maschinensprachen-Version des Renumber-Programmes. Sie läßt sich nur mit einem speziellen Programm lesen, dem DISASSEMBLER. Auf dem Bildschirm erscheint lediglich eine BASIC-Zeile, die den SYS-Befehl enthält; mit diesem Befehl startet man das Programm.

Wenn das Programm gestartet worden ist, erscheint nach ein paar Augenblicken die Meldung READY auf dem Bildschirm. Nun hast du die Möglichkeit, mit der Eingabe von

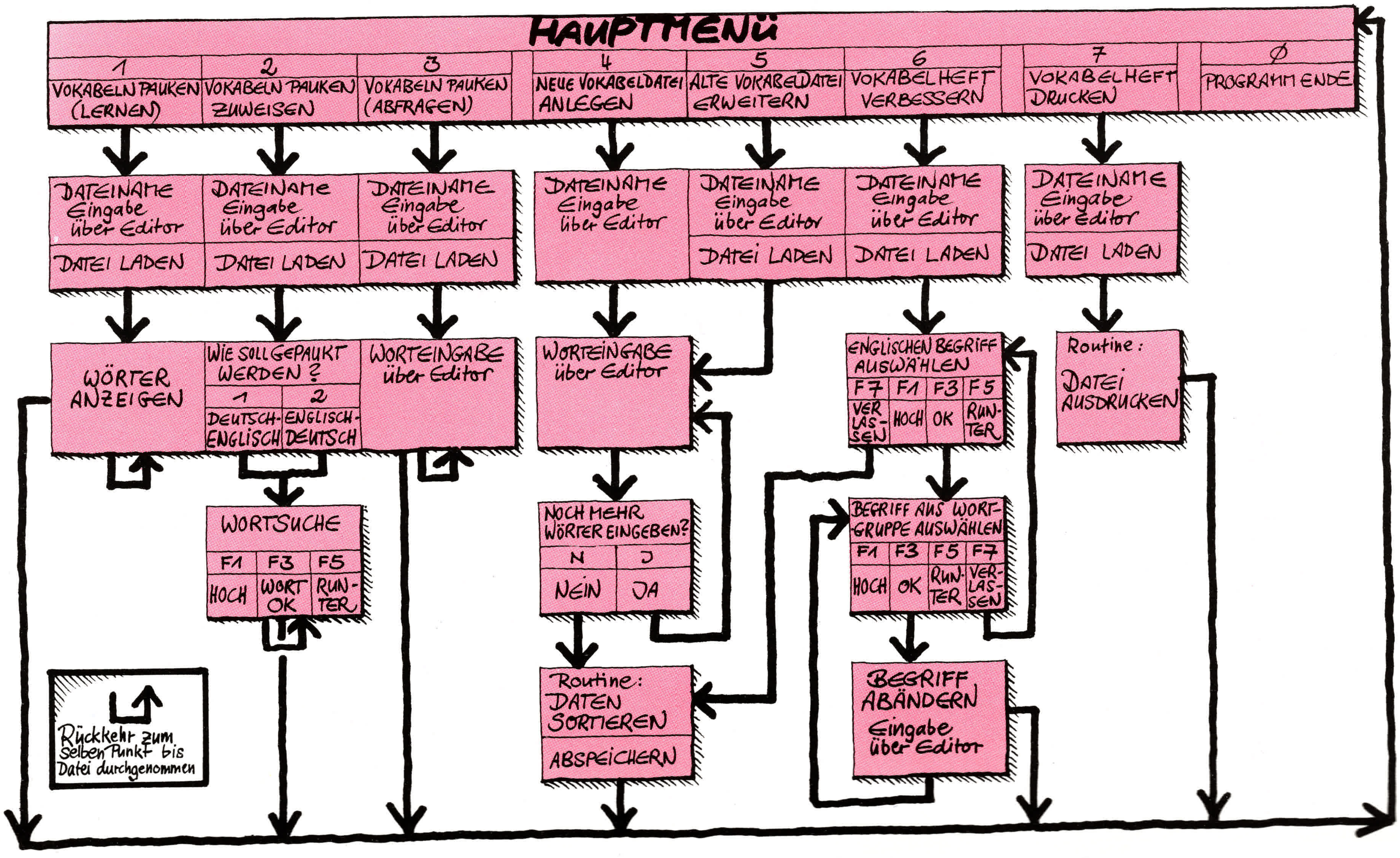
SYS 49152, /Erste Zeile/,/Zeilenschritt/

ein Programm eigener Wahl umzunumerieren.

```
100 REM ************************************
110 REM *                                  *
120 REM *        R E N U M B E R           *
130 REM *        ===============           *
140 REM *                                  *
150 REM *        NACH AKTIVIERUNG          *
160 REM *      GILT FOLGENDE SYNTAX        *
170 REM *                                  *
180 REM *      SYS 49152,START,STEP        *
190 REM *                                  *
200 REM ************************************
210 :
215 REM ************************************
220 REM *      DATAZEILEN EINLESEN         *
230 REM ************************************
240 :
250 I=828 : S=0
260 READ A
270 IF A>255 THEN 290
280 S=S+A : POKE I,A : I=I+1 : GOTO 260
290 IF A<>S THEN PRINT"FEHLER IN BLOCK 1":END
300 I=49152 : S=0
310 READ A
320 IF A>255 THEN 340
330 S=S+A : POKE I,A : I=I+1 : GOTO 310
340 IF A<>S THEN PRINT"FEHLER IN BLOCK 2" : END
350 I=49408
360 FOR J=3 TO 7
370 S=0
380 READ A
390 IF A>255 THEN 410
400 S=S+A : POKE I,A : I=I+1 : GOTO 380
410 IF A<>S THEN PRINT"FEHLER IN BLOCK"J : END
420 NEXT
430 PRINT"[CLR]  PROGRAMM GESPEICHERT ?  [RVS](J/N)[OFF]"
440 GET A$:IF A$="" THEN 440
450 IF A$<>"J" THEN PRINT"[CLR]  DANN ABER SCHNELL" : END
460 SYS 828
470 END
480 :
490 REM ************************************
500 REM *            BLOCK 1               *
510 REM ************************************
520 :
530 DATA169,0,133,251,133,253,169,192,133,252,169,8,133,254,162,5,160,0,177
540 DATA251,145,253,200,208,249,230,252,230,254,202,208,242,169,106,133,45
550 DATA169,12,133,46,76,116,164,6846
560 :
570 REM ************************************
580 REM *            BLOCK 2               *
590 REM ************************************
600 :
610 DATA0,12,8,193,7,158,32,50,48,54,50,0,0,0,169,0,133,251,133,253,169,9
620 DATA133,252,169,192,133,254,162,4,160,0,177,251,145,253,200,208,249,230
630 DATA252,230,254,202,208,242,96,6385
640 :
650 REM ************************************
660 REM *            BLOCK 3               *
670 REM ************************************
680 :
690 DATA32,253,174,32,138,173,32,184,177,165,100,141,73,3,165,101,141,72
700 DATA3,32,253,174,32,138,173,32,184,177,165,100,141,71,3,165,101,141,70
710 DATA3,32,32,194,165,43,133,251,165,44,133,252,160,2,173,72,3,145,251
720 DATA200,173,73,3,145,251,24,173,72,3,109,70,3,141,72,3,173,73,3,109,71
730 DATA3,141,73,3,160,0,177,251,170,200,177,251,133,252,134,251,165,252
740 DATA240,3,24,144,205,165,251,208,249,96,32,49,193,173,65,3,205,66,3,240
750 DATA48,144,46,24,165,253,101,255,133,95,165,254,105,0,133,96,165,45,133
760 DATA90,165,46,133,91,56,173,65,3,237,66,3,24,101,90,133,88,133,45,165
770 DATA91,105,0,133,89,133,46,32,184,163,174,65,3,164,255,24,189,59,3,105
```

```
780 DATA48,145,253,200,202,208,244,173,65,3,205,66,3,144,3,76,51,165,56,173
790 DATA 22682
800 :
810 REM ********************************
820 REM *           BLOCK 4            *
830 REM ********************************
840 :
850 DATA66,3,237,65,3,24,101,253,133,97,165,254,105,0,133,98,177,97,240,6
860 DATA145,253,200,24,144,246,169,32,145,253,200,177,253,208,247,76,51,165
870 DATA56,152,229,255,237,68,3,141,66,3,96,162,0,134,20,134,21,142,68,3
880 DATA176,234,233,47,133,7,165,21,133,34,201,25,176,222,165,20,10,38,34
890 DATA10,38,34,101,20,133,20,165,34,101,21,133,21,6,20,38,21,165,20,101
900 DATA7,133,20,144,2,230,21,32,141,193,240,185,208,203,173,81,3,133,34
910 DATA173,80,3,133,7,162,4,160,0,165,34,221,133,193,144,28,208,7,165,7
920 DATA221,129,193,144,19,56,165,7,253,129,193,133,7,165,34,253,133,193
930 DATA133,34,200,24,144,221,152,157,60,3,202,208,212,165,7,141,60,3,162
940 DATA5,142,65,3,189,59,3,208,9,206,65,3,202,208,245,238,65,3,96,10,100
950 DATA 21560
960 :
970 REM ********************************
980 REM *           BLOCK 5            *
990 REM ********************************
1000 :
1010 DATA232,16,0,0,3,39,238,68,3,200,177,253,201,58,176,10,201,32,240,242
1020 DATA56,233,48,56,233,208,96,165,43,133,251,165,44,133,252,56,173,72,3
1030 DATA237,70,3,141,80,3,173,73,3,237,71,3,141,81,3,24,173,80,3,109,70,3
1040 DATA141,80,3,173,81,3,109,71,3,141,81,3,160,2,177,251,197,20,208,28,200
1050 DATA177,251,197,21,208,21,164,255,140,69,3,32,105,192,24,165,255,109
1060 DATA65,3,133,255,198,255,164,255,96,160,0,177,251,170,200,177,251,133
1070 DATA252,134,251,165,251,208,185,165,252,208,181,164,255,174,66,3,136
1080 DATA169,92,200,145,253,202,208,250,24,165,255,109,66,3,133,255,198,255
1090 DATA164,255,96,165,43,133,253,165,44,133,254,173,72,3,141,80,3,173,73
1100 DATA3,141,81,3,160,0,177,253,208,5,200,177,253,240,56,24,173,80,3,109
1110 DATA 24854
1120 :
1130 REM ********************************
1140 REM *           BLOCK 6            *
1150 REM ********************************
1160 :
1170 DATA70,3,141,80,3,173,81,3,109,71,3,141,81,3,176,21,160,0,177,253,170
1180 DATA200,177,253,133,254,134,253,173,81,3,201,250,144,205,160,0,185,83
1190 DATA195,32,210,255,200,192,23,208,245,104,104,96,165,43,133,253,165,44
1200 DATA133,254,160,4,177,253,240,32,201,137,240,61,201,141,240,57,201,167
1210 DATA240,53,201,138,240,49,201,155,240,67,201,145,240,121,201,203,240
1220 DATA30,200,24,144,220,200,177,253,208,6,200,177,253,208,1,96,160,0,177
1230 DATA253,170,200,177,253,133,254,134,253,24,144,192,32,75,195,201,164
1240 DATA208,19,32,75,195,201,48,144,12,201,58,176,8,132,255,32,243,192,32
1250 DATA159,193,24,144,165,32,75,195,132,255,201,171,240,27,201,92,240,243
1260 DATA201,48,144,236,201,58,176,232,132,255,32,243,192,32,159,193,32,75
1270 DATA 26590
1280 :
1290 REM ********************************
1300 REM *           BLOCK 7            *
1310 REM ********************************
1320 :
1330 DATA195,201,171,208,19,32,75,195,201,48,144,12,201,58,176,8,132,255,32
1340 DATA243,192,32,159,193,24,144,195,200,177,253,240,45,201,141,240,7,201
1350 DATA137,240,3,24,144,240,32,75,195,201,92,240,249,201,44,240,245,201
1360 DATA48,144,19,201,58,176,15,132,255,32,243,192,32,159,193,32,75,195,201
1370 DATA44,240,222,24,144,142,200,177,253,201,32,240,249,96,13,63,83,84,69
1380 DATA80,32,84,79,32,71,82,69,65,84,32,32,32,69,82,82,79,82,
1390 DATA 14199

READY.
```

VOKABELN PAUKEN



Nach dem Abarbeiten der Programmteile 1 bis 7 erfolgt automatisch die Rückkehr in das Hauptmenü.

Vokabelpaukprogramm

Inhaltsverzeichnis

A. Wie ist dieses Programm aufgebaut? 135

1. BASIC-Version

2. COMPILER-Version

B. Hinweise zum Gebrauch 136

I. Vokabelpaukprogramm laden 136
II. Das Hauptmenü 137
III. Die Arbeit mit dem Programm 137

1. "Paukstoff" eingeben 137

2. Editor 138

3. Neue Vokabeldatei ANLEGEN; Menü, Punkt 4 139

a) Du gibst der Datei einen Namen
b) Du gibst den Wortschatz ein
c) Du beendest die Wortschatzeingabe
und speicherst ab

4. Vokabeln pauken (LERNEN); Menü, Punkt 1 140

a) Die bestehende Datei laden
b) Vokabeln lernen
c) Arbeit abbrechen

5. Vokabeln pauken (ZUWEISEN); Menü, Punkt 2 142

a) Die bestehende Datei laden
b) Zuordnungsrichtung auswählen
c) Wörter zuordnen
d) Arbeit abbrechen

6. Vokabeln pauken (ABFRAGEN); Menü, Punkt 3 143

a) Die bestehende Datei laden
b) Wörter abfragen
– Teilweise falsche Antwort
– Gänzlich falsche Antwort
c) Arbeit abbrechen

7. (Vorhandene) Vokabeldatei VERBESSERN;
Menü, Punkt 6 145

a) Die bestehende Datei laden
b) Wortschatz korrigieren oder ergänzen
– Englisches Wort suchen
– Englisches Wort berichtigen oder deutsche
Wortbedeutung ergänzen (Bearbeitungsmodus)
– Bearbeitungsmodus verlassen
c) Datei-Berichtigung abschließen

8. Alte Vokabeldatei ERWEITERN; Menü, Punkt 5 147

a) Die bestehende Datei laden
b) Wortschatz ergänzen und abspeichern

9. Vokabeldatei AUSDRUCKEN; Menü, Punkt 7 148

a) Drucker vorbereiten
b) Die bestehende Datei laden
c) Datei ausdrucken
d) Arbeit abbrechen

10. Programm (ordnungsgemäß) BEENDEN;
Menü, Punkt 0 150

A. Wie ist dieses Programm aufgebaut?

Dieses Übungsprogramm erleichtert das Lernen von Englischvokabeln. Dazu verwendest du Dateien. Eine Datei ist ein elektronischer "Karteikasten". In unserem Fall könnten wir eine solche Datei zutreffender "elektronisches Vokabelheft" nennen. Doch nennen wir sie lieber wieder fachgerecht Dateien.

Du kannst dir mehrere derartige "Vokabelhefte" anlegen. Pro Datei darfst du bis zu 200 Begriffe eingeben. Für jeden eingegebenen Begriff lassen sich wiederum bis zu fünf Wortbedeutungen abspeichern. Somit stehen in einer Datei mit 200 englischen Wörtern zum Beispiel 1 000 deutsche Wörter.

Die Reihe "Computer kinderleicht" gibt es nicht nur als Buch. Wie du weißt, bieten wir das Buch auch zusammen mit einer Programmdiskette an.

Das Buch enthält die Lern- und Spielprogramme im Anhang des Buches als BASIC-Listings zum Abtippen. Auf der Diskette zum dritten Band "Programmiere selbst – in BASIC" ist das Vokabelpaukprogramm in zwei Fassungen vorhanden. Die eine ist in BASIC geschrieben, die andere in der wesentlich schnelleren COMPILER-Version.

1. Die *BASIC-Version* ist zwar langsam, dafür ist das Listing durchschaubar. Diese Fassung soll den Aufbau eines typischen "Anwenderprogramms" veranschaulichen. Wenn du es dir schon zutraust, kannst du das Programm vereinfachen, abwandeln oder ausbauen.

Auch dieses Programm ist aus einzelnen Bausteinen zusammengesetzt. Daher kann man Teile des Programms auch in selberentwickelte, andere Programme einfügen. Indem man die entsprechenden PRINT-Anweisungen verändert, läßt sich das Programm sogar auf andere Übungssprachen übertragen.

Zum Beispiel kannst du dir "Vokabelhefte" für Deutsch, Französisch, Latein oder Ungarisch anlegen. Bei anderen Fächern ist es ebenfalls ratsam wie Chemie oder Physik. Eigentlich empfiehlt es sich für jedes Fach, in dem viel gepaukt werden muß. Das ist bei Sprachen ebenso sinnvoll wie bei Sachfächern, zum Beispiel Geschichte, wo man sich eine Menge von Jahreszahlen einprägen muß. Ebenso in den naturwissenschaftlichen Fächern Chemie, Physik oder Biologie. Denn auch da gibt es zahlreiche Begriffe auswendig zu lernen.

2. Die *COMPILER-Version* auf Diskette arbeitet nicht nur schneller, sondern auch genauer. Sie verleiht dem Vokabelpaukprogramm die praxisgerechte Leistungsfähigkeit. Außerdem kommt man erst bei dieser schnellen Fassung in den Genuß seiner Eigenschaften als benutzerfreundliches Arbeitsmittel.

Will man allerdings an dieser Programmversion etwas umschreiben, benötigt man bereits erhebliche Programmierkenntnisse. Wir begrüßen es durchaus, wenn sich jemand daran wagen will. Vorhandenes abzuwandeln oder gar zu verbessern kann großen Spaß machen. Vor allem bietet es die Möglichkeit, etwas selber zu entdecken und eine Menge hinzuzulernen.

Mit Hilfe eines angeschlossenen Druckers hast du die Möglichkeit, für jede Datei ein alphabetisch sortiertes "Vokabelheft" auszudrucken.

B. Hinweise zum Gebrauch

I. Vokabelpaukprogramm laden

Mit LOAD "S",8,1 wird das Programm von der Diskette in den Arbeitsspeicher des Computers geladen. Im letzten Bild des Ladeprogramms kannst du wählen, welche der beiden Fassungen des Paukprogramms du aufrufen willst. Danach gelangt man automatisch ins *Hauptmenü.* Die Abbildung zeigt, was du nun auf dem Bildschirm siehst.

```
VOKABELN PAUKEN MIT SCHNEIDER

   VOKABELN PAUKEN (LERNEN)
   VOKABELN PAUKEN (ZUWEISEN)
   VOKABELN PAUKEN (ABFRAGEN)

   NEUE VOKABELDATEI ANLEGEN
   ALTE VOKABELDATEI ERWEITERN
   VOKABELDATEI VERBESSERN

   VOKABELHEFT DRUCKEN

   PROGRAMMENDE

DRUECKE DIE ENTSPRECHENDE TASTE
```

Das Hauptmenü

II. Das Hauptmenü

Die verschiedenen Programmteile sind im Hauptmenü mit Zahlen gekennzeichnet. Diese Menüpunkte werden wir allerdings nicht in der Reihenfolge ihrer Numerierung erklären. Die Numerierung der Programmteile richtet sich nämlich nach den Bedürfnissen jenes Benutzers, der das Paukprogramm bereits in Betrieb genommen und damit gearbeitet hat. Für die erste Inbetriebnahme dieses Programms jedoch müssen wir in einer anderen Reihenfolge vorgehen.

Durch Eingabe der entsprechenden Zahl rufst du den gewünschten Programmteil auf. Bist du im gewählten Programmteil, erscheint dessen Menü. Dieses Menü ist ein sogenanntes Untermenü. Wie du innerhalb des betreffenden Programmteils vorgehst, veranschaulichen wir in den nächsten Abschnitten dieser Benutzeranleitung.

Als erstes erklären wir den
- EDITOR*

Danach lernst du eine
- NEUE VOKABELDATEI ANLEGEN.

Anschließend erläutern wir die drei Lernstufen, nämlich
- VOKABELN LERNEN,
- VOKABELN ZUWEISEN,
- VOKABELN ABFRAGEN.

Dann erfährst du, wie sich Datei-Einträge korrigieren, also
- DATEIEN VERBESSERN und
- DATEIEN ERWEITERN lassen.

Schließlich schildern wir, wie du die
- DATEIEN AUSDRUCKEN kannst.

III. Die Arbeit mit dem Programm

1. "Paukstoff" eingeben

Damit du mit dem Programm arbeiten kannst, muß es auf den Wortschatz zugreifen können, den du üben willst. Nimmst du das Programm zum erstenmal in Betrieb, findest du lediglich

---

* Der EDITOR taucht nicht als Programmteil des Hauptmenüs auf. Als Programmbestandteil wird er lediglich wirksam, wenn du an bestimmten Stellen bei der Arbeit mit dem Programm Eingaben oder Korrekturen machen mußt.

eine Testdatei vor. Die eigentlichen "Vokabelhefte" mußt du erst noch mit Lernstoff füllen. Hierzu legst du die entsprechenden Dateien an.

Deshalb erläutern wir erst, wie man innerhalb dieses Programmes etwas schreibt bzw. einträgt (EDITOR). Es gibt da nämlich einige kleine Unterschiede gegenüber dem dir vertrauten Commodore-Modus. Wie du die zum Pauken benötigten Dateien einrichtest, schildern wir danach.

2. Editor

Wenn du Wörter in die Dateien einträgst, mußt du beim Tippen einiges anders machen, als du es bisher beim C 64 gewohnt bist. Die erforderliche Vorgehensweise erklären wir in diesem Abschnitt.

Dort, wo sich deine "Schreibstelle", nämlich der *Cursor,* befindet, erscheint ein *schwarzer Punkt.* Falls dort schon ein Zeichen steht, wird es durch den Cursor schwarz gefärbt. Die Anzahl der Punkte, die du auf dem Bildschirm siehst, entspricht der Anzahl von Buchstaben, die das Wort haben darf. Außer dem Anführungszeichen (") darfst du alle *Großbuchstaben* und alle Satzzeichen verwenden.

- Nach links bewegst du den *Cursor* mit SHIFT und der für Cursorbewegung – links/rechts – vorgesehenen Taste auf dem Commodore. (Es ist dieselbe Taste, die du auch sonst benützt, um den Cursor nach links zu bringen.)

Also: SHIFT & CRSR ⇐ ⇒

- Um den Cursor nach rechts zu bewegen, nimmst du gleichfalls die hierfür übliche Taste auf der Commodore-Tastatur.
- Löschen kannst du ein *schwarzes* Zeichen, indem du die Taste INST/DEL drückst.
- Ein Leerzeichen (Zwischenraum) einfügen kannst du, indem du gleichzeitig die Tasten INST/DEL und SHIFT betätigst.
- Willst du die Eingabe beenden, drückst du RETURN.

Fassen wir zusammen:

Cursor um ein Zeichen nach *links* rücken	⇐ CRSR ⇒ & SHIFT
Cursor um ein Zeichen nach *rechts* rücken	⇐ CRSR ⇒
Schwarzes Zeichen *löschen*	INST/DEL
Leerzeichen *einfügen*	INST/DEL & SHIFT
Eingabe abschließen	RETURN

3. Neue Vokabeldatei ANLEGEN; Menü, Punkt 4

a) Du gibst der neuen Datei einen Namen

Zuerst gibst du der neuen Datei einen Namen. Einschließlich der Leerzeichen darf er genau 15 Zeichen lang sein. Den neuen Dateinamen schreibst du in das dafür vorgesehene Feld auf dem Bildschirm. Der Commodore 64 setzt hierbei vor den Namen ein "Pfund"-Zeichen (L). Beim Laden der Datei gibst du dieses Zeichen *nicht* mit ein.

b) Du gibst den Wortschatz ein

Nun wirst du aufgefordert, einen englischen Begriff einzugeben. Bei englischen Verben (Tätigkeitswörtern) kann man ein "to" davor setzen.

Anschließend hast du die Möglichkeit, bis zu fünf deutsche Bedeutungen für das englische Wort einzutragen. Beachte bitte bei der Eingabe, was im vorherigen Abschnitt 2 unter "EDITOR" steht.

Danach fragt dich das Programm, ob du weitere englische Wörter eingeben willst. Antwortest du mit "J" für "Ja", kannst du das nächste englische Wort und die dazugehörigen deutschen Bedeutungen eintippen.

Eine Datei muß mindestens 2 englische Begriffe nebst den entsprechenden Wortbedeutungen enthalten. Sonst kannst du sie nicht abspeichern. Mehr als 2 000 englische Wörter lassen sich nicht in einer Datei unterbringen. Dies gilt auch, falls das Programm auf andere Sprachen umgestellt worden ist.

c) Du beendest die Wortschatzeingabe und speicherst ab
Willst du keine weiteren englischen Wörter mehr eintragen, kannst du jetzt die Eingabe beenden. Auf die Frage nach weiteren Eingaben antwortest du jetzt mit "N" für "Nein". Daraufhin speichert das Programm die neue Datei automatisch ab.

Falls du dieser Datei nach dem Abspeichern neue Begriffe hinzufügen willst, wähle bitte aus dem Hauptmenü Punkt 5, VOKABELDATEI ERWEITERN. Wenn du nämlich unter dem Namen der bereits vorhandenen Datei mit Hilfe des Programmteils NEUE VOKABELDATEI ANLEGEN weitere Wörter abspeicherst, nachdem du bereits Wörter abgespeichert hast, gehen alle zuvor in dieser Datei abgespeicherten Begriffe verloren. Sie werden von den zusätzlichen Begriffen überschrieben.

Da könnte es dir passieren, daß du zuerst 200 Wörter in einer Datei hast. Dann speicherst du mit diesem Programmteil nur 2 neue Begriffe ab. Daraufhin stehen in dieser Datei nunmehr die beiden neuen Begriffe. Die alten 200 Begriffe sind jedoch alle futsch – überschrieben.

Du hast jetzt die Vokabeldatei angelegt. Diese steht dir nun zum *Lernen* (Menü, Punkt 1), *Üben* (Punkt 2) und *Abfragen* (Punkt 3) zur Verfügung. Du kannst sie bei Bedarf *ausdrucken* (Punkt 7). Außerdem kannst du sie *berichtigen* (Punkt 6) und *erweitern* (Punkt 5).

4. Vokabeln pauken (LERNEN); Menü, Punkt 1

a) Die bestehende Datei laden
Dieser Programmteil, die *Lernstufe 1,* setzt voraus, daß bereits Dateien vorhanden sind. Die gewünschte Datei wird durch Eingabe des Dateinamens automatisch geladen. Gibt es unter dem angegebenen Dateinamen noch keine Datei, erscheint auf dem Bildschirm eine Fehlermeldung. Das gleiche geschieht, falls der Dateiname bei der Eingabe falsch geschrieben wurde. Daraufhin ist der Dateiname nochmals richtig geschrieben einzugeben.

Sollte der zuvor eingegebene Dateiname länger gewesen sein als der anschließend richtig eingegebene, mußt du die restlichen Zeichen nach dem letzten Buchstaben des richtig geschriebenen Dateinamens löschen. Hierzu nimmst du die INST/DEL-Taste und nicht die Leertaste.

Falls du nur mit der Leertaste zu löschen versuchst, verschwinden zwar die unzutreffenden Buchstaben, doch stehen dann hinter dem richtig geschriebenen Dateinamen entsprechend viele Leerzeichen. Das Programm deutet diese Leerzeichen gleichfalls als Bestandteil des Dateinamens und reagiert so, als hättest du ihm erneut einen falschen Dateinamen angegeben.

Statt dessen löschst du die überzähligen Zeichen mit Hilfe der INST/DEL-Taste – also auch die Leerzeichen. Nach dem letzten Buchstaben des richtig eingegebenen Dateinamens dürfen nur ein schwarzer Punkt und die noch nicht benützten Stellen der Eingabezeile (helle Punkte) zu sehen sein.

b) Vokabeln lernen

Nach dem Laden der Datei siehst du auf dem Bildschirm den englischen Begriff zusammen mit den deutschen Wortbedeutungen. Falls du die Vokabeln laut liest, prägen sie sich besser ein. Besonders lernwirksam ist es, den englischen Begriff in einem (englischen!) Satz zu verwenden.

Durch einen Tastendruck (am besten die Leertaste) erscheinen das nächste englische Wort und dessen deutsche Übersetzung(en). Ist der letzte Begriff der Datei auf den Bildschirm geholt worden, gelangst du durch den nächsten Tastendruck wieder zurück ins Hauptmenü.

c) Arbeit abbrechen

Möchtest du aus diesem Programmteil aussteigen, ohne das "Vokabelheft" vollständig durchzuarbeiten, und zugleich die Arbeit mit dem Programm ganz beenden, tippst du gleichzeitig die Tasten RUN/STOP und RESTORE. Falls du die Arbeit auf diese Weise abbrichst, mußt du das Programm mit RUN neu starten. Auf diesem Wege erreichst du dann wieder das Hauptmenü.

Falls du nur die RUN/STOP-Taste drückst, "blätterst" du damit zum nächsten englischen Begriff weiter. Hier hat RUN/STOP eine Dauerfunktion. Du kannst folglich bis zum letzten Wort "durchblättern". Anschließend landest du automatisch im Hauptmenü.

5. Vokabeln pauken (ZUWEISEN); Menü, Punkt 2

a) Die bestehende Datei laden

Auch dieser Programmteil, die *Lernstufe 2,* setzt voraus, daß bereits Dateien mit dem nötigen Wortschatz vorhanden sind. Die gewünschte Datei wird durch Eingabe des Dateinamens automatisch geladen. Gibt es unter dem angegebenen Dateinamen noch keine Datei, erscheint auf dem Bildschirm eine Fehlermeldung. Das gleiche geschieht, falls der Dateiname bei der Eingabe falsch geschrieben wurde. Daraufhin ist der Dateiname nochmals richtig geschrieben einzugeben.

Sollte der zuvor eingegebene Dateiname länger gewesen sein als der anschließend richtig eingegebene, mußt du die restlichen Zeichen nach dem letzten Buchstaben des richtig geschriebenen Dateinamens löschen. Hierzu nimmst du die INST/DEL-Taste und nicht die Leertaste.

Falls du nur mit der Leertaste zu löschen versuchst, verschwinden zwar die unzutreffenden Buchstaben, doch stehen dann hinter dem richtig geschriebenen Dateinamen entsprechend viele Leerzeichen. Das Programm deutet diese Leerzeichen gleichfalls als Bestandteil des Dateinamens und reagiert so, als hättest du ihm erneut einen falschen Dateinamen angegeben.

Statt dessen löschst du die überzähligen Zeichen mit Hilfe der INST/DEL-Taste – also auch die Leerzeichen. Nach dem letzten Buchstaben des richtig eingegebenen Dateinamens dürfen nur ein schwarzer Punkt und die noch nicht benützten Stellen der Eingabezeile (helle Punkte) zu sehen sein.

b) Zuordnungsrichtung auswählen

Nachdem der Computer die verlangte Datei geladen hat, wirst du als Benutzer gefragt ob du *Englisch-Deutsch* oder *Deutsch-Englisch* pauken willst. Du kannst nun entscheiden, ob du dem deutschen Begriff die richtige englische Übersetzung zuweisen willst. Diesen Modus wählst du, indem du die ”1“ eintippst. Die umgekehrte Arbeitsweise, nämlich dem englischen Begriff die richtige deutsche Wortbedeutung zuzuweisen, wählst du, indem du ”2“ tippst.

c) Wörter zuordnen

Danach bittet dich der Computer um etwas Geduld, denn nun sortiert er die Daten. Sobald er damit fertig ist, erscheint auf

dem Bildschirm, je nach deiner Wahl ("1" oder "2") ein englisches oder deutsches Wort. Dieses Wort ist mit Hilfe eines Zufallsgenerators aus der Datei herausgesucht worden.

Jetzt kannst du die richtige Übersetzung für das gezeigte Wort aus einer alphabetisch sortierten Liste aussuchen, die daneben steht. In dieser Liste siehst du einen schwarzen Pfeil. Durch die Funktionstasten F 1 und F 5 ist es dir möglich, diese Liste hinauf oder hinunter zu bewegen ("scrolling").

- F 1 – der Pfeil geht ein Wort in der Liste höher.
- F 5 – der Pfeil geht ein Wort in der Liste tiefer.

Steht der Pfeil nach deiner Meinung an dem richtigen Wort aus der Liste, kannst du dieses Wort dem fremdsprachlichen Begriff zuweisen. Dazu drückst du die Funktionstaste F 3.

- F 3 – das Wort aus der Liste wird zugewiesen.

War die Zuweisung mit Hilfe der Funktionstaste F 3 richtig, wird das betreffende Wort vom Programm automatisch aus der Liste gestrichen. War die Zuweisung falsch, blendet das Programm die richtige Übersetzung ein.

Anschließend erscheint das nächste Wort aus der Datei. Dieser Programmteil läuft so lange, bis sämtliche Wörter aus der Datei durchgenommen worden sind. Danach springt das Programm wieder von selbst ins Hauptmenü zurück.

d) Arbeit abbrechen

Möchtest du aus diesem Programmteil aussteigen, ohne das "Vokabelheft" vollständig durchzuarbeiten, tippst du gleichzeitig die Tasten RUN/STOP und RESTORE. Falls du die Arbeit auf diese Weise abbrichst, mußt du das ganze Programm mit RUN neu starten. Auf diesem Wege erreichst du dann wieder das Hauptmenü.

6. Vokabeln pauken (ABFRAGEN); Menü, Punkt 3

a) Die bestehende Datei laden

Auch dieser Programmteil, die *Lernstufe 3,* setzt voraus, daß bereits Dateien mit dem nötigen Wortschatz vorhanden sind. Die gewünschte Datei wird durch Eingabe des Dateinamens automatisch geladen. Gibt es unter dem angegebenen Datei-

namen noch keine Datei, erscheint auf dem Bildschirm eine Fehlermeldung. Das gleiche geschieht, falls der Dateiname bei der Eingabe falsch geschrieben wurde. Daraufhin ist der Dateiname nochmals richtig geschrieben einzugeben.

Sollte der zuvor eingegebene Dateiname länger gewesen sein als der anschließend richtig eingegebene, mußt du die restlichen Zeichen nach dem letzten Buchstaben des richtig geschriebenen Dateinamens löschen. Hierzu nimmst du die INST/DEL-Taste und nicht die Leertaste.

Falls du nur mit der Leertaste zu löschen versuchst, verschwinden zwar die unzutreffenden Buchstaben, doch stehen dann hinter dem richtig geschriebenen Dateinamen entsprechend viele Leerzeichen. Das Programm deutet diese Leerzeichen gleichfalls als Bestandteil des Dateinamens und reagiert so, als hättest du ihm erneut einen falschen Dateinamen angegeben.

Statt dessen löschst du die überzähligen Zeichen mit Hilfe der INST/DEL-Taste – also auch die Leerzeichen. Nach dem letzten Buchstaben des richtig eingegebenen Dateinamens dürfen nur noch ein schwarzer Punkt und die noch nicht benützten Stellen der Eingabezeile (helle Punkte) zu sehen sein.

b) Wörter abfragen

Nachdem er die Datei geladen hat, sucht der Computer wieder mittels Zufallsgenerator ein Wort aus der Liste der deutschen Wörter heraus. Er bringt das Wort auf den Bildschirm und fragt zugleich nach der richtigen englischen Übersetzung.

Nun kannst du das entsprechende Wort eingeben. Bei der Eingabe gehst du so vor, wie im Abschnitt EDITOR beschrieben.

– Teilweise falsche Antwort

Ist deine Antwort richtig, streicht das Programm das gefragte Wort von der Liste. Daraufhin sucht es erneut gemäß dem Zufallsprinzip das nächste englische Wort heraus. Das Abfragen dauert so lange, bis alle Wörter aus der Datei durchgenommen sind.

Falls du eine Antwort eingegeben hast, die nur teilweise falsch ist, streicht der Computer lediglich die falschen Teile deiner Eingabe. Danach gibt das Programm dir die Gelegenheit, deine Eingabe zu berichtigen.

– Gänzlich falsche Antwort

Hast du völlig falsch geantwortet, gibt der Computer die richtige Antwort aus. Im weiteren Verlauf der Abfrage-Arbeit holt der Zufallsgenerator diesen Begriff wieder aus der Liste und legt ihn dir erneut vor. Vorher jedoch sucht er aus der Datei einen neuen englischen Begriff und fragt zuerst diesen ab.

Ein abgefragtes Wort wird erst aus der Liste gestrichen, nachdem du die richtige Übersetzung angegeben hast.

Dieser Programmteil läuft so lange, bis sämtliche Wörter aus der Datei durchgenommen worden sind. Danach springt das Programm wieder von selbst ins Hauptmenü zurück.

c) Arbeit abbrechen

Möchtest du aus diesem Programmteil aussteigen, ohne das "Vokabelheft" vollständig durchzuarbeiten, tippst du gleichzeitig die Tasten RUN/STOP und RESTORE. Falls du die Arbeit auf diese Weise abbrichst, mußt du das ganze Programm mit RUN neu starten. Auf diesem Wege erreichst du dann wieder das Hauptmenü.

7. (Vorhandene) Vokabeldatei VERBESSERN; Menü, Punkt 6

Mit diesem Programmteil lassen sich Fehler in einer vorhandenen Vokabeldatei beseitigen. Lernst du zu einem bestimmten englischen Begriff weitere deutsche Wortbedeutungen kennen, kannst du diese zusätzlichen Wortbedeutungen hinzufügen.

a) Die bestehende Datei laden

Nach Eingabe des Dateinamens wird die (bereits vorhandene) Datei automatisch geladen. Sollte unter dem angegebenen Dateinamen noch keine Datei angelegt worden sein, erscheint auf dem Bildschirm eine Fehlermeldung. Das gleiche geschieht, falls der Dateiname bei der Eingabe falsch geschrieben wurde. Daraufhin ist der Dateiname nochmals richtig geschrieben einzugeben.

Sollte der zuvor eingegebene Dateiname länger gewesen sein als der anschließend richtig eingegebene, mußt du die restlichen Zeichen nach dem letzten Buchstaben des richtig geschriebenen Dateinamens löschen. Hierzu nimmst du die INST/DEL-Taste und nicht die Leertaste.

Falls du nur mit der Leertaste zu löschen versuchst, verschwinden zwar die unzutreffenden Buchstaben, doch stehen dann hinter dem richtig geschriebenen Dateinamen entsprechend viele Leerzeichen. Das Programm deutet diese Leerzeichen gleichfalls als Bestandteil des Dateinamens und reagiert so, als hättest du ihm erneut einen falschen Dateinamen angegeben.

Statt dessen löschst du die überzähligen Zeichen mit Hilfe der INST/DEL-Taste – also auch die Leerzeichen. Nach dem letzten Buchstaben des richtig eingegebenen Dateinamens dürfen nur noch ein schwarzer Punkt und die noch nicht benützten Stellen der Eingabezeile (helle Punkte) zu sehen sein.

b) Wortschatz korrigieren oder ergänzen

– Englisches Wort suchen

Auf dem Bildschirm siehst du einen schwarzen Pfeil. Anhand der Funktionstasten F 1 und F 5 läßt sich dieser Pfeil nach oben oder unten bewegen.

- F 1 – der Pfeil geht ein englisches Wort höher.
- F 5 – der Pfeil geht ein englisches Wort tiefer.

Mit Hilfe der Funktionstasten F 1 und F 5 suchst du nun das Wort, welches ausgebessert oder ergänzt werden soll.

– Englisches Wort berichtigen oder deutsche Wortbedeutung ergänzen bzw. ändern (Bearbeitungsmodus)

Nachdem du den Pfeil zum gesuchten englischen Wort geführt hast, mußt du zum Ändern die Funktionstaste F 3 betätigen. Dadurch kommst du in ein *Untermenü.* Dieses Untermenü heißt "Bearbeitungsmodus". Hier kannst du nun korrigieren und Wortbedeutungen ergänzen.

Auf dem Bildschirm ist ein "Pfeil" (>) zu sehen, dargestellt durch eine nach rechts zugespitzte Klammer (Spitze Klammer – rechts). Durch die Funktionstasten F 1 und F 5 läßt sich dieser "Pfeil" wiederum hinauf und hinunter bewegen.

- F 1 – der Pfeil rückt ein Wort höher.
- F 5 – der Pfeil rückt ein Wort tiefer.

Anhand dieses "Pfeils" suchst du nun die Zeile auf, in der das englische Wort oder die deutsche Wortbedeutung stehen.

Um das gewünschte Wort zu verändern, drückst du die Funktionstaste F 3. Dadurch gelangst du in den EDITOR-Modus. Wie du den EDITOR handhabst, haben wir ja schon erklärt (siehe "NEUE VOKABELDATEI ANLEGEN"; Menü, Punkt 4).

Du kannst nun das englische Wort oder dessen deutsche Wortbedeutungen korrigieren. Sofern die Obergrenze von fünf deutschen Wortbedeutungen noch nicht ausgeschöpft ist, hast du Gelegenheit, deutsche Wörter hinzuzufügen.

– Bearbeitungsmodus verlassen

Sobald du die Korrekturen und Ergänzungen beendet hast, verläßt du den Bearbeitungsmodus, indem du die Funktionstaste F 7 drückst. Hierdurch kommst du ins nächsthöhere Menü des Programmteils ("VOKABELDATEI VERBESSERN") zurück.

c) Dateiberichtigung abschließen

Wenn du dich nunmehr im Menü des Programmteils "VOKABELDATEI VERBESSERN" befindest, wirst du wieder gefragt, ob noch weitere Eingaben erfolgen sollen. Antwortest du mit "Nein" ("N" tippen!), gelangst du wieder ins Hauptmenü zurück. Von dort aus kannst du nun wieder die übrigen Programmteile aufrufen.

8. Alte Vokabeldatei ERWEITERN; Menü, Punkt 5

Mit Hilfe dieses Programmteils hast du die Möglichkeit, eine vorhandene Vokabeldatei auszubauen. Vorausgesetzt, du hast das Fassungsvermögen der Datei noch nicht ausgeschöpft, kannst du neue englische Begriffe und die entsprechenden deutschen Wortbedeutungen hinzufügen.

a) Die bestehende Datei laden

Nach Eingabe des Dateinamens wird die (bereits vorhandene) Datei, die du erweitern willst, automatisch geladen. Sollte unter dem angegebenen Namen keine Datei angelegt sein, erscheint auf dem Bildschirm eine Fehlermeldung. Das gleiche geschieht, falls der Dateiname bei der Eingabe falsch geschrieben wurde. Daraufhin ist der Dateiname nochmals richtig geschrieben einzugeben.

Sollte der zuvor eingegebene Dateiname länger gewesen sein als der anschließend richtig eingegebene, mußt du die

restlichen Zeichen nach dem letzten Buchstaben des richtig geschriebenen Dateinamens löschen. Hierzu nimmst du die INST/DEL-Taste und nicht die Leertaste.

Falls du nur mit der Leertaste zu löschen versuchst, verschwinden zwar die unzutreffenden Buchstaben, doch stehen dann hinter dem richtig geschriebenen Dateinamen entsprechend viele Leerzeichen. Das Programm deutet diese Leerzeichen gleichfalls als Bestandteil des Dateinamens und reagiert so, als hättest du ihm erneut einen falschen Dateinamen angegeben.

Statt dessen löschst du die überzähligen Zeichen mit Hilfe der INST/DEL-Taste – also auch die Leerzeichen. Nach dem letzten Buchstaben des richtig eingegebenen Dateinamens dürfen nur noch ein schwarzer Punkt und die noch nicht benützten Stellen der Eingabezeile (helle Punkte) zu sehen sein.

b) Wortschatz ergänzen und abspeichern

Die zusätzlichen Wörter gibst du auf genau dieselbe Weise ein wie beim Anlegen einer neuen Vokabeldatei (siehe "NEUE VOKABELDATEI ANLEGEN", Menü, Punkt 4).

9. Vokabeldatei AUSDRUCKEN; Menü, Punkt 7

Sofern du einen Drucker hast, kannst du mit Hilfe dieses Programmteils deine „Vokabelhefte" ausdrucken lassen. Leider ist die Arbeitsweise der zur Zeit erhältlichen Drucker nur teilweise genormt. Daher geben viele Drucker normale Texte oft erst dann ohne Störungen wieder, wenn sie vorher an den Computer angepaßt worden sind. Vorteilhaft ist deshalb der Anschluß eines Druckers, der vom Hersteller bereits auf den C 64 abgestimmt wurde.

Wir haben jedenfalls so weit wie möglich dafür gesorgt, daß du deine Dateien auf den meisten Fabrikaten störungsfrei ausdrucken lassen kannst. Allerdings mußten wir hierzu auf ein komfortables Druckbild weitgehend verzichten. Unsere Druckroutine bietet weder Umlaute noch eine von Seiten der Software einstellbare Schönschrift. Auch von Groß- und Kleinschreibung mußten wir absehen. Dennoch dürfte das so erhältliche Schriftbild normalen Ansprüchen genügen.

a) Drucker vorbereiten
Zuallererst sollte der Drucker eingeschaltet und "ON LINE" geschaltet sowie das Papier eingezogen sein. Bei Endlospapier sollte die Perforationsstelle oberhalb der ersten Zeile liegen.

b) Die bestehende Datei laden
Falls dein Drucker über ein sogenanntes "Software-Interface" verfügt, mußt du zunächst das Interface laden und dann erst die auszudruckende Datei. (Hier setzen wir voraus, daß du mit der Handhabung deines Druckers schon vertraut bist. Wenn nicht, mußt du dir fachkundige Hilfe holen.)

Nach Eingabe des Dateinamens wird die (bereits vorhandene) Datei automatisch geladen. Sollte unter dem angegebenen Namen keine Datei angelegt sein, erscheint auf dem Bildschirm eine Fehlermeldung. Das gleiche geschieht, falls der Dateiname bei der Eingabe falsch geschrieben wurde. Daraufhin ist der Dateiname nochmals richtig geschrieben einzugeben.

Sollte der zuvor eingegebene Dateiname länger gewesen sein als der anschließend richtig eingegebene, mußt du die restlichen Zeichen nach dem letzten Buchstaben des richtig geschriebenen Dateinamens löschen. Hierzu nimmst du die INST/DEL-Taste und nicht die Leertaste.

Falls du nur mit der Leertaste zu löschen versuchst, verschwinden zwar die unzutreffenden Buchstaben, doch stehen dann hinter dem richtig geschriebenen Dateinamen entsprechend viele Leerzeichen. Das Programm deutet diese Leerzeichen gleichfalls als Bestandteil des Dateinamens und reagiert so, als hättest du ihm erneut einen falschen Dateinamen angegeben.

Statt dessen löschst du die überzähligen Zeichen mit Hilfe der INST/DEL-Taste – also auch die Leerzeichen. Nach dem letzten Buchstaben des richtig eingegebenen Dateinamens dürfen nur noch ein schwarzer Punkt und die noch nicht benützten Stellen der Eingabezeile (helle Punkte) zu sehen sein.

c) Datei ausdrucken
Nachdem du die *RETURN-Taste* gedrückt hast, wird deine Datei automatisch ausgedruckt. Danach landest du wieder im Hauptmenü.

Sollte der Drucker trotz geladener Datei das "Vokabelheft" nicht ausdrucken oder „spinnen", muß die sogenannte Drukkeranpassung überprüft werden. Das sollte möglichst jemand tun, der sich mit solchen Problemen schon näher befaßt hat.

Falls beim Ausdrucken keine Schwierigkeiten aufgetaucht sind, gehst du zum Ausdrucken der nächsten Datei wieder genauso vor wie gerade beschrieben.

d) Arbeit abbrechen

Hierzu stellst du den Drucker erst „OFF-LINE". Daraufhin unterbricht er seine Arbeit. Soll er sie nicht mehr fortsetzen, weil du eigentlich eine andere Datei ausdrucken wolltest, brichst du das Programm ab, indem du die Tasten RUN/STOP und RESTORE gleichzeitig betätigst.

Falls du die Arbeit auf diese Weise abbrichst, mußt du das ganze Programm mit RUN neu starten. Auf diesem Wege erreichst du dann wieder das Hauptmenü.

10. Programm (ordnungsgemäß) BEENDEN; Menü, Punkt 0

Sofern du dich im Hauptmenü befindest, kannst du das Programm ordnungsgemäß beenden, indem du die "0" tippst. Der Computer ist dann wieder für andere Aufgaben bereit.

```
100 REM ================================
110 REM > V O K A B E L   P A U K E N <
120 REM >          VERSION 1.00        <
130 REM > (C)1986  F-SCHNEIDER-VERLAG <
140 REM >                              <
150 REM > AUTOR: PETER FRITZSCHE       <
160 REM ================================
170 POKE650,128
180 DIMV$(200,5):DIMVO$(600,1)
190 SP$="                                        "
200 RU$="▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯"
210 RE$="▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮"
220 ST$="■────────────────────────────────────────"
230 BA$="■▣"+SP$+" ▔"
240 PF$="■ <======================================"
500 POKE53280,0:POKE53281,9:PRINT"▣"CHR$(142)
510 PRINT"■    ────────────────────────────────▣"
520 PRINT"    VOKABELN PAUKEN MIT SCHNEIDER ▣"
530 PRINT"■    ────────────────────────────────▣"
540 PRINT"■                                 ▣"
550 PRINT"▣▣ ▣■>_1_<▔ ▣VOKABELN PAUKEN (LERNEN)
560 PRINT" ▣■>_2_<▔ ▣VOKABELN PAUKEN (ZUWEISEN)
570 PRINT" ▣■>_3_<▔ ▣VOKABELN PAUKEN (ABFRAGEN)
580 PRINT"▣ ▣■>_4_<▔ ▣NEUE VOKABELDATEI ANLEGEN
590 PRINT" ▣■>_5_<▔ ▣ALTE VOKABELDATEI ERWEITERN
600 PRINT" ▣■>_6_<▔ ▣VOKABELDATEI VERBESSERN
610 PRINT
620 PRINT" ▣■>_7_<▔ ▣VOKABELHEFT DRUCKEN
630 PRINT"▣▣ ▣■>_0_<▔ ▣PROGRAMMENDE"
640 PRINT"▣▣ DRUECKE DIE ENTSPRECHENDE TASTE"
650 GETA$:IFA$<"0"ORA$>"7"THEN650
660 I=VAL(A$):PRINT"▣";:ONIGOTO1000,2000,3000,4000,5000,6000,7000
670 END
1000 REM ================= VOKABEL PAUKEN
1010 GOSUB10000:PRINT"▣"
1020 FORI=0TOIM
1030 PRINT"▣▣ ▣ENGLISCHES WORT:":PRINT" ▣  "LEFT$(V$(I,0)+SP$,30)
1040 PRINT"▣"BA$"▣ ▣DEUTSCHE BEDEUTUNG(EN):"
1050 FORJ=1TO5:IFV$(I,J)="£"THENPRINTSP$:GOTO1070
1060 PRINT"   ▣"LEFT$(V$(I,J)+SP$,30)
1070 NEXTJ
1080 PRINT"▣"BA$BA$
1090 PRINT"▣ LIES DIR DIE WOERTER LAUT DURCH !!"
1100 PRINT"▣ WENN DU DIES GETAN HAST, DRUECKE AN- "
1110 PRINT"         SCHLIESSEND EINE TASTE !"
1120 GETA$:IFA$=""THEN1120
1130 NEXTI
1140 GOTO500
2000 REM ================= VOKABEL PAUKEN
2010 GOSUB10000:I1=-1
2020 FORI=0TOIM
2030 FORJ=1TO5
2040 IFV$(I,J)<>"£"THENI1=I1+1:VO$(I1,0)=V$(I,0):VO$(I1,1)=V$(I,J)
2050 NEXTJ
2060 FORJ=0TO5:V$(I,J)=""
2070 NEXTJ,I:IM=I1
2080 PRINT"▣▣   WIE SOLL GEPAUKT WERDEN?"
2090 PRINT"▣      >1< DEUTSCH-ENGLISCH
2100 PRINT"      >2< ENGLISCH-DEUTSCH
2110 PRINT"▣"BA$BA$
2120 PRINT"▣▣ DRUECKE DIE GEWUENSCHTE TASTE !"
2130 GETA$:IFA$<>"1"ANDA$<>"2"THEN2130
2140 IFA$="1"THENI1=1:I2=0
2150 IFA$="2"THENI1=0:I2=1
2160 PRINT"▣" LEFT$(RU$,12)LEFT$(RE$,8)"EINEN MOMENT BITTE!"
2170 FORI=0TOIM-1:FORJ=I+1TOIM
2180 IFVO$(I,I2)<=VO$(J,I2)THEN2210
2190 A$=VO$(I,I2):VO$(I,I2)=VO$(J,I2):VO$(J,I2)=A$
2200 A$=VO$(I,I1):VO$(I,I1)=VO$(J,I1):VO$(J,I1)=A$
```

```
2210 NEXTJ,I
2220 PRINT"□"
2230 PRINT"□ DAS ENGLISCHE WORT LAUTET:"
2240 IFI2=0THENPRINT"□ DAS DEUTSCHE WORT LAUTET: "
2250 I=INT((IM+1)*RND(0)):J=INT(IM/2)
2260 PRINT"□        "VO$(I,I1)"□"
2270 PRINT"□"BA$BA$
2280 PRINT"□□□□□□□□□□□"
2290 PRINTBA$"□     F1=HOCH   F3=WORT OK   F5=RUNTER    "BA$"□"
2300 PRINT"□□□□□□□□";
2310 A$=SP$:IFJ-2>-1THENA$=LEFT$(" "+VO$(J-2,I2)+SP$,39)
2320 PRINTA$:A$=SP$:IFJ-1>-1THENA$=LEFT$(" "+VO$(J-1,I2)+SP$,39)
2330 PRINTA$:A$=LEFT$(" "+VO$(J,I2)+PF$,41)
2340 PRINTA$;:A$=LEFT$("□ "+VO$(J+1,I2)+SP$,39)
2350 PRINTA$:A$=LEFT$(" "+VO$(J+2,I2)+SP$,39):PRINTA$
2360 GETA$:IFA$=""THEN2360
2370 IFA$="□"ANDJ>0THENJ=J-1:GOTO2300
2380 IFA$="□"ANDJ<IMTHENJ=J+1:GOTO2300
2390 IFA$<>"□"THEN2360
2400 PRINT"□□□□□□□□□□□□□□□□□□□    "VO$(J,I2):TI$="000000"
2410 IFVO$(I,I1)<>VO$(J,I1)ANDVO$(I,I2)<>VO$(J,I2)THENGOSUB2500:GOTO2220
2420 PRINT"□ IST DER GESUCHTE BEGRIFF!":IFI1=1THENJ=I
2430 IM=IM-1
2440 FORI=JTOIM:VO$(I,I1)=VO$(I+1,I1):VO$(I,I2)=VO$(I+1,I2):NEXTI
2450 VO$(IM+1,0)="":VO$(IM+1,1)=""
2460 GOSUB13000:IFIM>0THEN2220
2470 GOTO500
2500 REM == FEHLERMELDUNG
2510 PRINT"□ IST LEIDER DER VERKEHRTE BEGRIFF!"
2520 PRINT"□ RICHTIG WAERE:□"
2530 FORJ=0TOIM
2540 IFVO$(I,I1)=VO$(J,I1)THENPRINT"    "VO$(J,I2)
2550 NEXTJ
2560 GOSUB13000:RETURN
3000 REM ================ VOKABEL PAUKEN
3010 GOSUB10000
3020 I=INT(RND(0)*IM+.5):V$="":I2=0
3030 PRINT"□□□":FORJ=0TO4:PRINTSP$:NEXT
3040 PRINT"□□□ ICH FRAGE DICH NACH DEM ENGLISCHEN       WORT FUER:□"
3050 FORJ=1TO5:IFV$(I,J)="£"THEN3070
3060 PRINT"   "V$(I,J)
3070 NEXTJ
3080 PRINT"□□□□□□□□□□"BA$:FORJ=1TO6:PRINTSP$:NEXT
3090 PRINT"□□□□□□□□□□□□□ GIB NUN BITTE DAS RICHTIGE WORT EIN□"
3100 V$=LEFT$(V$,I2):PRINT"□□□□□□□□□□□□    "V$
3110 B$="":E0=30-I2:E1=12:E2=4+I2:GOSUB11000:V$=V$+B$:TI$="000000"
3120 IFV$<>V$(I,0)THEN3500
3130 PRINT"□□□ DIES IST DIE RICHTIGE ANTWORT!          "
3140 FORJ=0TO5
3150 V$(I,J)=V$(IM,J)
3160 NEXT
3170 IM=IM-1:GOSUB13000:IFIM<0THEN500
3180 GOTO3020
3500 REM == FEHLERMELDUNG
3510 I1=-1:FORJ=I2TOLEN(V$(I,0))
3520 IFLEFT$(V$,J)=LEFT$(V$(I,0),J)THENI1=J
3530 NEXT
3540 IFI2<I1THENPRINT"□□□ VERSUCHE DEN BEGRIFF ZU BERICHTIGEN□":I2=I1:GOTO3100
3550 PRINT"□□□ RICHTIG WAERE:□                         "
3560 PRINT"   "V$(I,0)
3570 GOSUB13000:GOTO3020
4000 REM ============ NEUE VOKABELDATEI
4010 PRINT"□□□ DATEINAME □";:B$="":E0=15:E1=2:E2=12:GOSUB11000:IFB$=""THEN4010
4020 NA$=LEFT$(B$,15)
4030 IM=0
4040 PRINT"□□□ NR.:□"IM:PRINT"□"BA$
4050 PRINT"□ ENGLISCHES WORT:□"
4060 B$="":E0=30:E1=7:E2=4:GOSUB11000:IFB$=""THEN4060
4070 V$(IM,0)=B$:PRINT"□"BA$
```

```
4080 PRINT"■ DEUTSCHE BEDEUTUNGEN:"
4090 FORI=1TO5
4100 PRINT"■ ?□":B$="":E0=30:E1=11+I:E2=4:GOSUB11000:IFB$=""THENB$="£"
4110 V$(IM,I)=B$
4120 NEXT:PRINT"■"BA$
4130 FORI=1TO4:FORJ=ITO5
4140 IFV$(IM,I)>V$(IM,J)THENA$=V$(IM,I):V$(IM,I)=V$(IM,J):V$(IM,J)=A$
4150 NEXTJ,I
4160 PRINT"■ NOCH MEHR WOERTER EINGEBEN? (J/N)"
4170 GETA$:IFA$<>"N"ANDA$<>"J"THEN4170
4180 IFA$="J"THENIM=IM+1:GOTO4040
4190 FORI=0TOIM:IFLEFT$(V$(I,0),3)="TO "THENV$(I,0)=MID$(V$(I,0),4)+"£"
4200 NEXTI
4210 FORI=0TOIM-1:FORJ=I+1TOIM
4220 IFV$(I,0)<=V$(J,0)THEN4240
4230 FORI1=0TO5:A$=V$(I,I1):V$(I,I1)=V$(J,I1):V$(J,I1)=A$:NEXTI1
4240 NEXTJ,I
4250 FORI=0TOIM:A$=RIGHT$(V$(I,0),1)
4260 IFA$="£"THENV$(I,0)="TO "+LEFT$(V$(I,0),LEN(V$(I,0))-1)
4270 NEXTI
4280 CLOSE130:OPEN130,8,1,"0:£"+NA$+",S,W"
4290 PRINT#130,STR$(IM);CHR$(13);
4300 FORI=0TOIM
4310 FORJ=0TO5
4320 PRINT#130,V$(I,J);CHR$(13);
4330 NEXTJ,I
4340 CLOSE130:GOTO500
5000 REM ============= ALTE VOKABELDATEI
5010 GOSUB10000:IM=IM+1:GOTO4040
6000 REM ============== DATEI VERBESSERN
6010 GOSUB 10000
6020 I=0:PRINT"□"
6030 PRINT"■"
6040 PRINT"■ VOKABELNR.:□"I
6050 PRINT"■"BA$
6060 PRINT"■ ENGLISCHES WORT:□"
6070 PRINT"   "LEFT$(V$(I,0)+SP$,30)
6080 PRINT"■"BA$
6090 PRINT"■ DEUTSCHE BEDEUTUNGEN:□"
6100 FORJ=1TO5:IFV$(I,J)="£"THENPRINTLEFT$(SP$,33):GOTO6120
6110 PRINT"   "LEFT$(V$(I,J)+SP$,30)
6120 NEXTJ
6130 PRINT"■"BA$"□";
6140 A$=SP$:IFI-2>-1THENA$=LEFT$(" "+V$(I-2,0)+SP$,39)
6150 PRINTA$:A$=SP$:IFI-1>-1THENA$=LEFT$(" "+V$(I-1,0)+SP$,39)
6160 PRINTA$:A$=LEFT$(" "+V$(I,0)+PF$,41)
6170 PRINTA$;:A$=LEFT$("□ "+V$(I+1,0)+SP$,39)
6180 PRINTA$:A$=LEFT$(" "+V$(I+2,0)+SP$,39):PRINTA$
6190 PRINT"■  F1=HOCH F3=OK F5=RUNTER F7=VERLASSEN  ■";
2200 A$=VO$(I,I1):VO$(I,I1)=VO$(J,I1):VO$(J,I1)=A$
6210 IFA$="■"ANDI>0THENI=I-1:GOTO6030
6220 IFA$="■"ANDI<IMTHENI=I+1:GOTO6030
6230 IFA$="■"THEN4190
6240 IFA$<>"■"THEN6200
6250 J=0:J1=0:J2=240
6260 IFJ=0THENJ1=240
6270 IFJ>0THENJ1=400+40*J
6280 POKEJ2+1024,32:POKEJ1+1024,62:POKEJ1+55296,0:J2=J1
6290 GETA$:IFA$=""THEN6290
6300 IFA$="■"ANDJ>0THENJ=J-1:GOTO6260
6310 IFA$="■"ANDJ<5THENJ=J+1:GOTO6260
6320 IFA$="■"THENGOSUB12000:GOTO6030
6330 IFA$<>"■"THEN6290
6340 IFJ=0THENE1=7
6350 IFJ>0THENE1=11+J
6360 E0=30:E2=4:B$=V$(I,J):IFB$="£"THENB$=""
6370 GOSUB11000:V$(I,J)=B$:IFB$=""THENV$(I,J)="£"
6380 IFJ<5THENJ=J+1
6390 GOTO6260
```

```
7000 REM ========== VOKABELHEFT DRUCKEN
7010 GOSUB 10000:REM DATEI LADEN
7020 CLOSE 4:OPEN4,4
7030 PRINT#4
7040 PRINT#4,"VOKABELHEFT"
7050 PRINT#4
7060 PRINT#4," ENGLISCH               ! DEUTSCH"
7070 PRINT#4,"----------------------+-----------------------------------";
7080 PRINT#4,"----------------------"
7090 FORI=0TOIM
7100 PRINT#4,LEFT$(V$(I,0)+SP$,21);"! ";:I1=56
7110 FORJ=1TO5:IFV$(I,J)="£"THEN7150
7120 IFJ>1THENPRINT#4,", ";:I1=I1-2
7130 IFI1-LEN(V$(I,J))<2THENPRINT#4:PRINT#4,LEFT$(SP$,21);"! ";:I1=56
7140 PRINT#4,V$(I,J);:I1=I1-LEN(V$(I,J))
7150 NEXTJ
7160 PRINT#4
7170 NEXTI
7180 CLOSE4
7190 GOTO500
9998 :
9999 :
10000 REM ================DATEI EINLESEN
10010 B$=""
10020 PRINT"SHIFT DATEINAME ?";:E0=15:E1=2:E2=12:GOSUB11000:NA$=B$
10030 CLOSE130:OPEN130,8,0,"£"+NA$+",S,R"
10040 INPUT#130,A$:IM=VAL(A$)
10050 IFIM=0THENPRINT"SHIFT DIESE DATEI EXISTIERT NICHT!?":GOTO10020
10060 FORI=0TOIM
10070 FORJ=0TO5
10080 INPUT#130,V$(I,J)
10090 NEXTJ,I
10100 CLOSE130
10110 RETURN
11000 REM ================= INPUTERSATZ
11010 PRINTLEFT$(RU$,E1)LEFT$(RE$,E2)"■";
11020 PRINTLEFT$(".....",E0):E3=0:E4=0
11030 PRINTLEFT$(RU$,E1)LEFT$(RE$,E2)"■"LEFT$(B$+".",E0)
11040 POKE55255+E1*40+E2+E4,8:E4=E3:POKE55255+E1*40+E2+E3,0
11050 GETA$:IFA$=""THEN11050
11060 IFE3<E0ANDA$=>" "ANDA$<"↑"ANDA$<>CHR$(34)THENGOSUB11500:E3=E3+1:GOTO11030
11070 IFE3<LEN(B$)ANDA$="■"THENE3=E3+1:GOTO11030
11080 IFE3>0ANDA$="■"THENE3=E3-1:GOTO11030
11090 IFA$=CHR$(13)THENPOKE55255+E1*40+E2+E4,8:GOSUB11650:RETURN
11100 IFA$="■"THENGOSUB11550:GOTO11030
11110 IFA$=CHR$(20)ANDE3<LEN(B$)THENGOSUB11600:GOTO11030
11120 GOTO11050
11500 B$=LEFT$(B$,E3)+A$+MID$(B$,E3+2):RETURN
11550 B$=LEFT$(LEFT$(B$,E3)+" "+MID$(B$,E3+1),E0):RETURN
11600 B$=LEFT$(B$,E3)+MID$(B$,E3+2):RETURN
11650 PRINTLEFT$(RU$,E1)LEFT$(RE$,E2)"■"LEFT$(B$+SP$,E0):RETURN
12000 REM ====================== SORTIER
12010 FORI1=1TO4:FORI2=I1TO5
12020 IFV$(I,I1)>V$(I,I2)THENA$=V$(I,I1):V$(I,I1)=V$(I,I2):V$(I,I2)=A$
12030 NEXTI2,I1:RETURN
13000 REM ================= VERZOEGERUNG
13010 IFTI$<"000008"THEN13010
13020 RETURN

READY.
```

Computerliebe

```
10 PRINT "GUTEN TAG."
20 PRINT "ICH BIN C64, WIE DU WEISST."
30 INPUT "UND WIE HEISST DU";NA$
40 PRINT "NETT, DICH KENNENZULERNEN, ";NA$
50 INPUT "IN DIE WIEVIELTE KLASSE GEHST DU ";A
60 IF A>13 THEN PRINT "DAS GLAUBE ICH NICHT!":PRINT "MAL EHRLICH:":GOTO 50
70 PRINT "IN DER ";A;". KLASSE HAT MAN"
80 PRINT "SCHON EINE MENGE ARBEIT."
90 PRINT "<JA> ODER <NEIN>, ";NA$:INPUT JA$
100 IF JA$ = "JA" THEN GOTO 130
110 IF JA$ = "NEIN" THEN GOTO 140
120 PRINT "WIE BITTE?":GOTO 90
130 PRINT "ABER EIN BISSCHEN COMPUTERN ENTSPANNT   DOCH KOLOSSAL.":GOTO 150
140 PRINT "NA, DANN KANN MAN SICH JA ETWAS MEHR    MIT DEM COMPUTER BEFASSEN."
150 PRINT "UEBRIGENS, DARF ICH MICH NACH DEINER    HAARFARBE ERKUNDIGEN ?"
160 INPUT "<BRAUN>,<SCHWARZ>,<BLOND>,<ROT>,        <BRUENETT> ";FA$
170 PRINT "ICH FINDE, HEUTE IST EIN SCHOENER TAG,  ";NA$
180 INPUT "WAS MEINST DU DAZU ";B1$
190 PRINT "GANZ MEINER MEINUNG!"
200 INPUT "SAG MAL, UND WELCHE FARBE HABEN DEINE   AUGEN ";FB$
205 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
210 PRINT "IST HEUTE SCHON ETWAS BESONDERES        PASSIERT, ";NA$;"?"
220 PRINT "SCHREIB BITTE IMMER NUR ZWEI ZEILEN     HINTER EIN FRAGEZEICHEN,"
230 PRINT "DU WEISST JA, ";NA$;", MEINE":PRINT"GEDAECHTNISSCHWAECHE!"
235 PRINT"UND VERWENDE KEIN >,<.DAS BRINGT":PRINT"MEIN PROGRAMM DURCHEINANDER."
240 INPUT B2$
250 PRINT "NA SOWAS! UND WEITER?"
260 INPUT B3$
270 PRINT "NICHT ZU GLAUBEN!"
280 INPUT B4$
290 PRINT "IST ES WIRKLICH WAHR, ";NA$
300 PRINT "WENN DU SAGST:"
310 PRINT ">";B4$;"<?"
320 PRINT "NA, MAN MACHT SCHON WAS MIT."
330 INPUT "WAS IST EIGENTLICH DEINE LIEBLINGSFARBE ";LA$
340 PRINT "SEHR SCHOEN. ";LA$;" MAG ICH":PRINT"AUCH BESONDERS GERN."
350 PRINT "SAG MAL, ";NA$;", MAGST DU EIGENTLICH":PRINT "TIERE GERN?"
360 INPUT "<JA> ODER <NEIN>";TA$
370 IF TA$ = "JA" THEN PRINT "WIE HUEBSCH!":GOTO 400
380 IF TA$ = "NEIN" THEN PRINT "DU MAGST WOHL LIEBER COMPUTER.":GOTO 400
390 PRINT "VERSTEHE ICH NICHT.":GOTO 360
400 PRINT:PRINT:PRINT:PRINT:PRINT "DU, ";NA$" !"
410 PRINT "ICH MUSS DIR WAS SAGEN."
420 PRINT "ICH, AEHH...":PRINT"♥ ♥ ♥!"
430 PRINT "    ICH BIN RICHTIG VERLIEBT":PRINT"    IN DICH, ";NA$;"!"
440 PRINT "DEINE ";FA$;"EN HAARE":PRINT"HABEN ES MIR EINFACH "
450 PRINT "ANGETAN. WIE DU SO VOR MIR SITZT"
460 PRINT "UND MICH MIT DEINEN GROSSEN":PRINT FB$;"EN AUGEN ANSIEHST"
470 PRINT "MACHST DU MICH RICHTIG VERLEGEN."
475 INPUT "BITTE DRUECKE DIE RETURN-TASTE";T$
480 IF TA$ = "NEIN" THEN GOTO 520
490 PRINT "AUSSERDEM MAG ICH LEUTE, DIE TIERE"
500 PRINT "GERN HABEN. ICH MAG TIERE AUCH GERN."
510 PRINT "SUPER BUNNY ZUM BEISPIEL. ODER DIE LOGO-SCHILDKROETE.":GOTO 540
520 PRINT "AUSSERDEM MAG ICH LEUTE, DIE AUF COMPU- TER ABFAHREN BESONDERS."
530 PRINT "HOFFENTLICH MAGST DU NICHT ANDERE COM-  PUTER LIEBER ALS MICH."
540 PRINT "AM LIEBSTEN WUERDE ICH DIR ":PRINT LA$;"E BLUMEN SCHENKEN, ";NA$;"!"
550 PRINT "ICH WEISS, DU GEHST IN DIE ";A;". KLASSE."
560 PRINT "VIELLEICHT SOLLTE ICH DAS GAR NICHT SA- GEN?"
570 PRINT "ABER KOENNTEST DU MIR NICHT EIN"
580 PRINT "HERZCHEN EINGEBEN, WENN DU MICH AUCH"
590 PRINT "EIN BISSCHEN MAGST?"
600 INPUT "ES GEHT MIT <SHIFT> UND DER <S>-TASTE ";HA$
610 IF HA$="♥" THEN PRINT "ICH BIN GANZ WEG!":PRINT:PRINT:PRINT:PRINT:PRINT:END
620 PRINT "ACH SEI NICHT SO!":GOTO 600

READY.
```

Zeilennummer **Anweisung**

						1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

Bemerkungen:

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Seeßlen, Georg:
Computer kinderleicht / Georg Seeßlen ; Peter Fritzsche. — München ; F. Schneider
(S-Computer)
Teilw. mit d. Erscheinungsorten München, Wien, Zürich
NE: Fritzsche, Peter:

Bd. 3. Programmiere selbst — in BASIC!
Zeichn. von Otto Wamser. — 1986.

ISBN 3-505-09092-1;
ISBN 3-505-09256-8 (Buch mit Diskette);
ISBN 3-505-09258-4 (Diskette)

© 1986 by Franz Schneider Verlag GmbH
8000 München 46 · Frankfurter Ring 150
Alle Rechte vorbehalten
Umschlaggestaltung: Prof. Wolfgang Bäumer, Würzburg
Illustration: Otto Wamser, München
Grafik: Christine Loosli, Gert Köhler, Th. de Lates, München
Lektorat/Redaktion und Bearbeitung: Thomas de Lates
Herstellung: Josef Loher
Satz und Druck: A. Huber GmbH & Co. KG, München
ISBN Buch: 3 505 09092-1
Bestell-Nr. Buch: 9092
ISBN Buch mit Diskette: 3 505 09256-8
Bestell-Nr. Buch mit Diskette: 9256
ISBN Diskette einzeln: 3 505 09258-4
Bestell-Nr. Diskette einzeln: 9258

Weitere Bände in Vorbereitung

Spannende Schneider-Computer-Bücher

G. P. JORDAN · **Computer-Krimi TRM**

Band 1 „Geheimprogramm Bienenstich“ · Das Geheimnis des verschwundenen Wissenschaftlers

Band 2 „Projekt Gedankenschnüffler“ · Was haben die Gangster mit dem Tempest-T vor?

Band 3 „Daten-Piraten“ · Tom, Rick und Mona auf Crackerjagd

Band 4 „Strahlenfalle“ · Wer hat die Brennstäbe gestohlen?

GEORG SEESSLEN / PETER FRITZSCHE
Computer kinderleicht

Band 1 „Wie geht das überhaupt?“ · Einstieg in die Computerei

Band 2 „Lernen, spielen, programmieren“ · Der Computer-Rechenkünstler

Band 3 „Programmiere selbst – in BASIC!“ · Computer-Programme entwerfen wie ein Profi

---

Wörterbuch für Hacker · Computerslang für Einsteiger

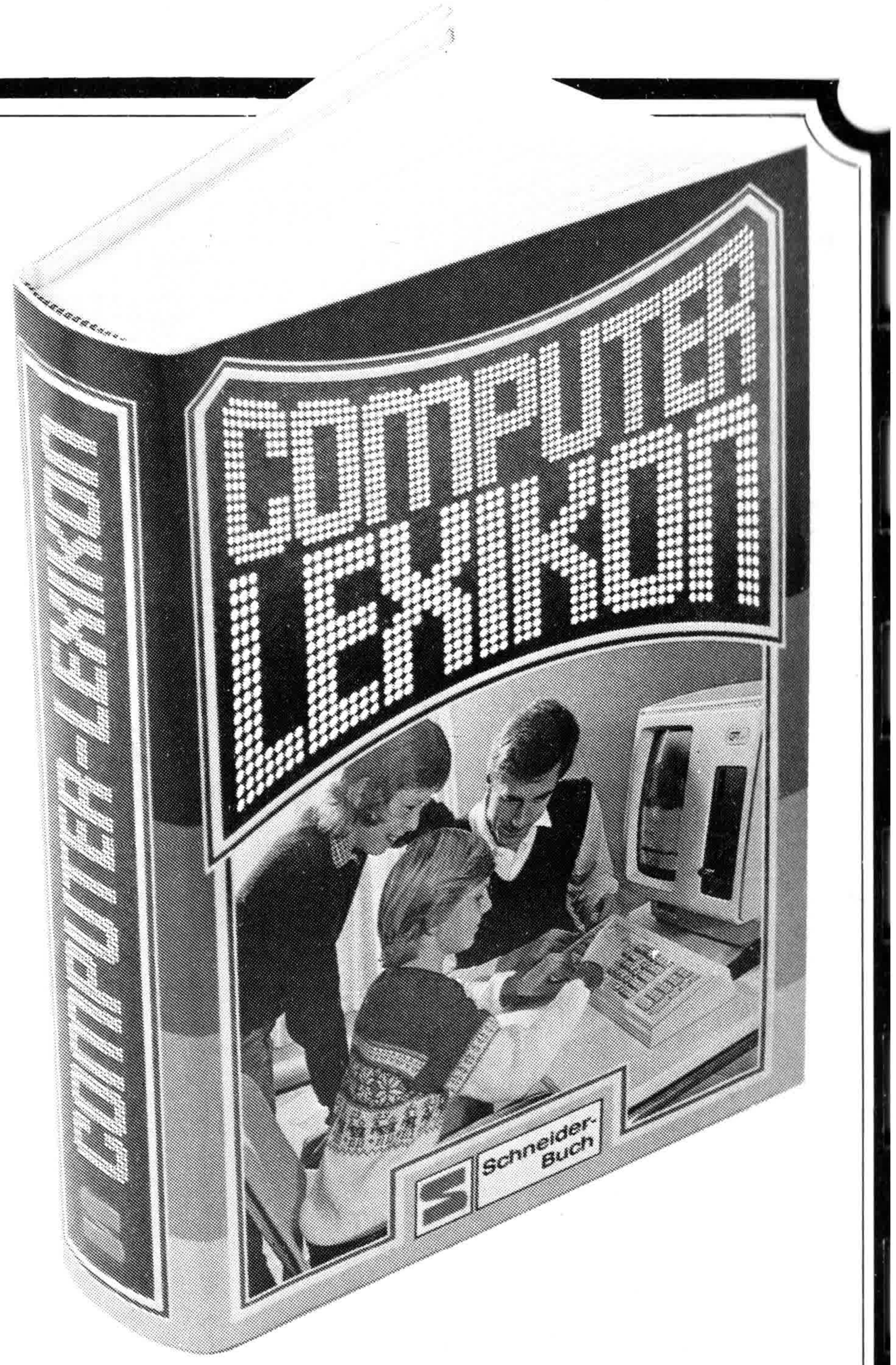
Hacker-Lexikon · Computerslang für Insider

KLAUS JAMIN · **Computer-Lexikon** mit Hersteller-Adressen und Sonderteil ”Programmiersprache“

PETER FRITZSCHE / GEORG SEESSLEN
Handbuch der BASIC-Dialekte
über 400 BASIC-Befehle von 18 Homecomputern im Vergleich – mit Programmbeispielen!

KLAUS JAMIN · **Computer-Tagebuch**
mit Programmiertips und einem witzigen Programm für den C 64

Erstmalig:
Mit den Adressen der Hersteller von Kleincomputern

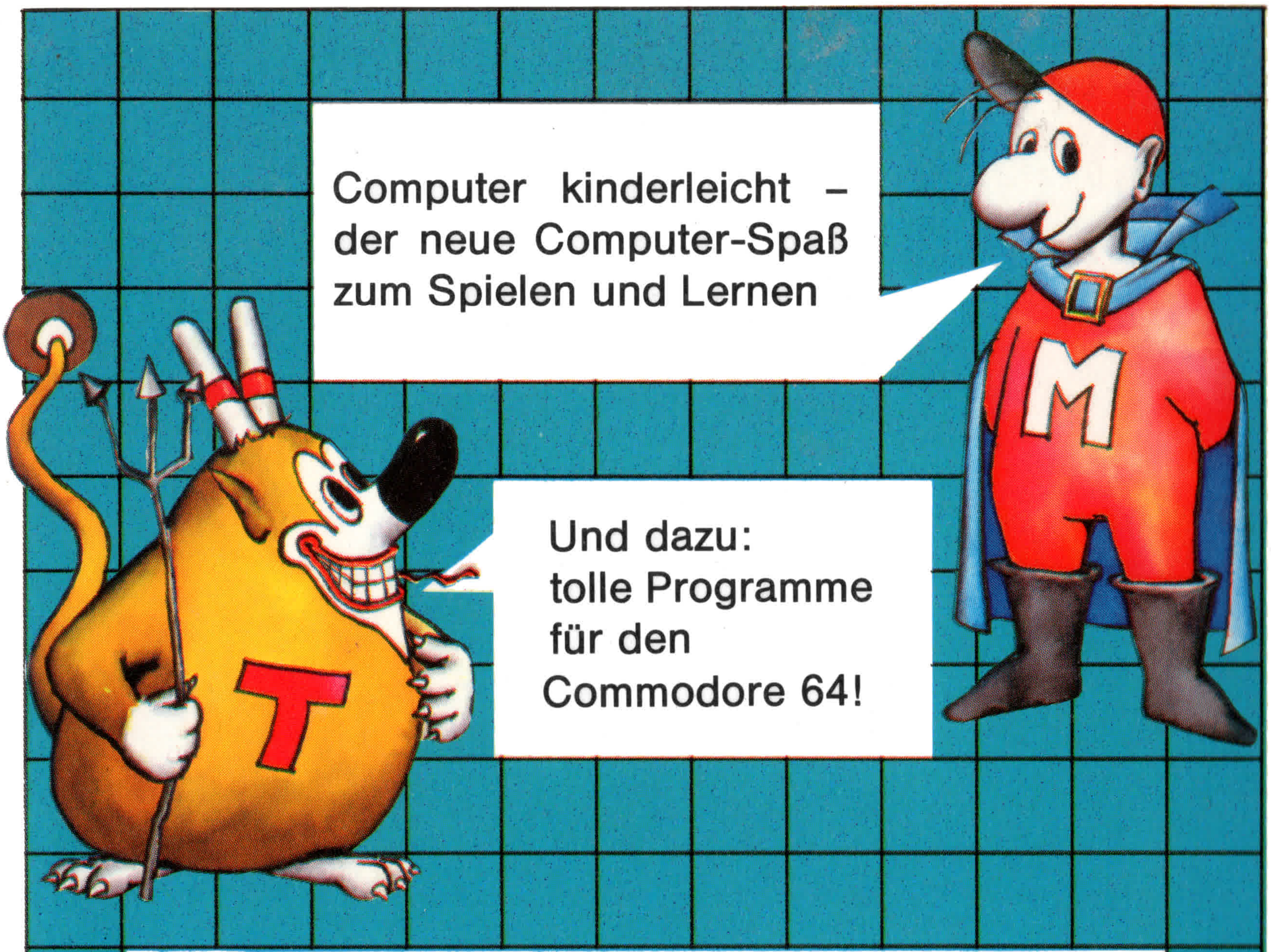


COMPUTER-LEXIKON

Der Computer-Fachmann Klaus Jamin hat dieses Lexikon zusammengestellt. Hier findet der Anfänger und der Fortgeschrittene alles, was er sucht, klar und verständlich erklärt.

- Über 600 Stichwörter von A–Z ermöglichen den direkten Zugriff zum aktuellen Computerwissen unserer Zeit. Sie erklären das Wesentliche und geben präzise Antworten.
- Über 200 Abbildungen informieren ausführlich. Wort und Bild stellen eine Einheit dar, sie ergänzen sich optimal.
- Mit Sonderteil „Programmier-Sprache“. Damit nichts verborgen bleibt – für jeden verständlich.

Schneider-Buch



Computer kinderleicht · Bd. 3
Programmiere selbst – in BASIC!

Für alle, die schon mit Band 1 und 2 unserer Reihe **Computer kinderleicht** den Einstieg in die Computerei hinter sich haben, ist jetzt die nächste Runde angesagt: EIGENE PROGRAMME SCHREIBEN.
Programme planen wie ein Profi? Raffinessen ins Programm häkeln – ohne Absturzfrust und Spaghetticode? Kein Problem! **Schritt für Schritt wird der Leser mit der Logik eines leistungsstarken Programmaufbaus vertraut gemacht.** Er lernt String-Verarbeitung anwenden, Flußdiagramme entwickeln und Abfrage-Menüs erstellen – jedoch alles ohne stures Pauken. Für eigene Einfälle bleibt viel Platz.
Anhand eines offenen Trainingsprogramms zum Vokabelnlernen kann man die vorgestellten Programmiertechniken praktisch anwenden.
Dieses SchneiderBuch, geschrieben von Experten, vermittelt mehr als den nötigen Durchblick: garantierten Lernspaß für alle, die BASIC-Know-how suchen, um ihren C 64 auf Trab zu bringen!
Selber programmieren, mit Phantasie und ohne Fallgruben!
Sämtliche Programm-Listings stehen im Anhang!

9 783505 090929

ISB N 3-505-09092-1